Springer Handbook of Robotics

2nd Edition
With 1375 Figures and 109 Tables

Bruno Siciliano, Oussama Khatib (Eds.)

Springer

Avidbots Ex1016
Page 1 of 2154

Editors
Bruno Siciliano
University of Naples Federico II
Department of Electrical Engineering and Information Technology
Naples, Italy
siciliano@unina.it

Oussama Khatib
Stanford University
Department of Computer Science
Artificial Intelligence Laboratory
Stanford, USA
khatib@cs.stanford.edu

ISBN: 978-3-319-32550-7 e-ISBN: 978-3-319-32552-1
DOI 10.1007/978-3-319-32552-1
Library of Congress Control Number: 2016937424

© Springer-Verlag Berlin Heidelberg 2016

This Springer imprint is published by Springer Nature
The registered company is Springer-Verlag GmbH Berlin Heidelberg

Foreword

Bernard Roth
Professor of
Mechanical Engineering
Stanford University

My first introduction to robotics came via a phone call in 1964. The caller was Fred Terman, the author of the world-famous *Radio Engineer's Handbook*, who was at the time Provost of Stanford University. Dr. Terman informed me that a computer science professor, John McCarthy, had just been awarded a large research grant, part of which required the development of computer-controlled manipulators. Someone had suggested to Terman that it would be prudent if the mathematically oriented McCarthy had some contact with mechanical designers. Since I was the only one on the Stanford faculty whose specialty was mechanism design, Terman decided to phone me, even though we had never met and I was a young assistant professor fresh out of graduate school with only 2 years at Stanford.

Dr. Terman's phone call led me to a close association with John McCarthy and the Stanford Artificial Intelligence Laboratory (SAIL) that he founded. Robotics became one of the pillars of my entire academic career, and I have maintained my interest in teaching and researching the subject through to the present day.

The modern history of robotic manipulation dates from the late 1940s when servoed arms were developed in connection with master–slave manipulator systems used to protect technicians handling nuclear materials. Developments in this area have continued to the present day. However, in the early 1960s there was very little academic or commercial activity in robotics. The first academic activity was the thesis of H. A. Ernst, in 1961, at MIT. He used a slave arm equipped with touch sensors, and ran it under computer control. The idea in his study was to use the information from the touch sensors to guide the arm.

This was followed by the SAIL project and a similar project started by Professor Marvin Minsky at MIT, which were the only sizeable academic ventures into robotics at that time. There were a few attempts at commercial manipulators, primarily in connection with part production in the automotive industry. In the USA there were two different manipulator designs that were being experimented with in the auto industry; one came from American Machine and Foundry (AMF) and the other from Unimation, Inc.

There were also a few mechanical devices developed as hand, leg, and arm prosthetics, and, a bit later, some exoskeletal devices to enhance human performance. In those days there were no microprocessors. So, these devices were either without computer control, or tethered to a remote so-called minicomputer, or even a mainframe computer.

Initially, some in the computer science community felt that computers were powerful enough to control any mechanical device and make it perform satisfactorily. We quickly learned that this was not to be the case. We started on a twofold track. One was to develop particular devices for SAIL, so that hardware demonstrations and proof-of-concept systems were available for the fledgling robotics community to experiment with. The other track, which was more or less moonlighted from the work at SAIL, was the development of a basic mechanical science of robotics. I had a strong feeling that a meaningful science could be developed, and that it would be best to think in terms of general concepts rather than concentrate exclusively on particular devices.

Fortuitously, it turned out that the two tracks supported each other very naturally and, most importantly, the right students were interested in doing their research in this area. Hardware developments proved to be specific examples of more general concepts, and the students were able to develop both the hardware and the theory.

Originally, we purchased an arm in order to get started quickly. A group at Rancho Los Amigos Hospital, in Los Angeles, was selling a tongue-switch-controlled motor-driven exoskeleton arm to assist patients without muscular control of their arms. We purchased one of these, and connected it to a time-shared PDP-6 computer. The device was named *Butterfingers*; it was our first experimental robot. Several films demonstrating visual feedback control, block stacking tasks, and obstacle avoidance were made with *Butterfingers* as the star performer.

The first manipulator that we designed on our own was known simply as the *Hydraulic Arm*. As its name implies, it was powered by hydraulics. The idea was to build a very fast arm. We designed special rotary actuators, and the arm worked well. It became the experimental platform for testing the first ever dynamic analysis and time-optimal control of a robotic arm. However, its use was limited since the design speeds were much faster than required due to the limitations

of the computational, planning, and sensing capabilities that were common at that time.

We made an attempt to develop a truly digital arm. This led to a snake-like structure named the *Orm* (the Norwegian word for snake.) The *Orm* had several stages, each with an array of inflatable pneumatic actuators that were either fully extended or fully contracted. The basic idea was that, even though only a finite number of positions in the workspace could be reached, these would be sufficient if there were a large number of positions. A small prototype proof-of-concept *Orm* was developed. It led to the realization that this type of arm would not really serve the SAIL community.

The first truly functional arm from our group was designed by Victor Scheinman, who was a graduate student at the time. It was the very successful *Stanford Arm*, of which over ten copies were made as research tools to be used in various university, government, and industrial laboratories. The arm had six independently driven joints; all driven by computer-controlled servoed, DC electric motors. One joint was telescoping (prismatic) and the other five were rotary (revolute).

Whereas the geometry of *Butterfingers* required an iterative solution of the inverse kinematics, the geometric configuration of the *Stanford Arm* was chosen so that the inverse kinematics could be programmed in any easy-to-use time-efficient closed form. Furthermore, the mechanical design was specifically made to be compatible with the limitations inherent in time-share computer control. Various end-effectors could be attached to act as hands. On our version, the hand was in the form of a vise-grip jaw, with two sliding fingers driven by a servoed actuator (hence, a true seventh degree of freedom). It also had a specially designed six-axis wrist force sensor. Victor Scheinman went on to develop other important robots: the first was a small humanoid arm with six revolute joints. The original design was paid for by Marvin Minsky at the MIT AI Lab. Scheinman founded Vicarm, a small company, and produced copies of this arm and the *Stanford Arm* for other labs. Vicarm later became the West Coast Division of Unimation, Inc., where Scheinman designed the *PUMA* manipulator under General Motors sponsorship through Unimation. Later, for a company called Automatix, Scheinman developed the novel *Robot World* multirobot system. After Scheinman left Unimation, his colleagues Brian Carlisle and Bruce Shimano reorganized Unimation's West Coast Division into Adept, Inc., which to this day is the largest US manufacturer of assembly robots.

Quickly, the modern trend of carefully detailed mechanical and electronic design, optimized software, and complete system integration became the norm; to this day, this combination represents the hallmark of most highly regarded robotic devices. This is the basic concept behind *mechatronic*, a word conied in Japan as a concatenation of the words mechanics and electronics. Mechatronics that relies on computation is the essence of the technology inherent in robotics as we know it today.

As robotics developed around the world, a large number of people started working on various aspects, and specific subspecialties developed. The first big division was between people working on manipulators and those working on vision systems. Early on, vision systems seemed to hold more promise than any other method for giving robots information about their environment.

The idea was to have a television camera capture pictures of objects in the environment, and then use algorithms that allowed the computer images of the pictures to be analyzed, so as to infer required information about location, orientation, and other properties of objects. The initial successes with image systems were in problems dealing with positioning blocks, solving object manipulation problems, and reading assembly drawings. It was felt that vision held potential for use in robotic systems in connection with factory automation and space exploration. This led to research into software that would allow vision systems to recognize machine parts (particularly partially occluded parts, as occurred in the so-called *bin-picking* problems) and ragged-shaped rocks.

After the ability to *see* and move objects became established, the next logical need had to do with planning a sequence of events to accomplish a complex task. This led to the development of planning as an important branch in robotics. Making fixed plans for a known fixed environment is relatively straightforward. However, in robotics, one of the challenges is to let the robot discover its environment, and to modify its actions when the environment changes unexpectedly due to errors or unplanned events. Some early landmark studies in this area were carried out using a vehicle named *Shakey*, which, starting in 1966, was developed by Charlie Rosen's group at the Stanford Research Institute (now called SRI). *Shakey* had a TV camera, a triangulating range finder, bump sensors, and was connected to DEC PDP-10 and PDP-15 computers via radio and video links.

Shakey was the first mobile robot to reason about its actions. It used programs that gave it the ability for independent perception, world modeling, and action generation. Low-level action routines took care of simple moving, turning, and route planning. Intermediate-

level actions combined the low-level ones in ways that accomplished more complex tasks. The highest level programs could make and execute plans to achieve high-level goals supplied by a user.

Vision is very useful for navigation, locating objects, and determining their relative positions and orientation. However, it is usually not sufficient for assembling parts or working with robots where there are environmental constraining forces. This led to the need to measure the forces and torques generated by the environment, on a robot, and to use these measurements to control the robot's actions. For many years, force-controlled manipulation became one of the main topics of study at SAIL, and several other labs around the world. The use of force control in industrial practice has always lagged the research developments in this area. This seems to be due to the fact that, while a high level of force control is very useful for general manipulation issues, specific problems in very restricted industrial environments can often be handled with limited, or no, force control.

In the 1970s, specialized areas of study such as walking machines, hands, automated vehicles, sensor integration, and design for hostile environments began to develop rapidly. Today there are a large number of different specialties studied under the heading of robotics. Some of these specialties are classical engineering subject areas within which results have been developed that have been particularized to the types of machines called robots. Examples here are kinematics, dynamics, controls, machine design, topology, and trajectory planning. Each of these subjects has a long history predating the study of robotics; yet each has been an area of in-depth robotics research in order to develop its special character in regard to robotic-type systems and applications. In doing this specialized development, researchers have enriched the classical subjects by increasing both their content and scope.

At the same time that the theory was being developed, there was a parallel, although somewhat separate, growth of industrial robotics. Strong commercial development occurred in Japan and Europe, and there was also continued growth in the USA. Industrial associations were formed (the Japan Robot Association was formed in March 1971, and the Robotic Industries Association (RIA) was founded in 1974 in the USA) and trade shows, together with application-oriented technical sessions, were introduced and held on a regular basis. The most important were the International Symposium on Industrial Robots, the Conference on Industrial Robot Technology (now called the International Conference on Industrial Robot Technology), and the RIA annual trade show, which is now called the International Robots and Vision Show and Conference.

The first regular series of conferences emphasizing research, rather than the industrial, aspects of robotics, was inaugurated in 1973. It was sponsored jointly by the International Center for Mechanical Sciences (CISM), based in Udine, Italy, and the International Federation for the Theory of Mechanisms and Machines (IFToMM). (Although IFToMM is still used, its meaning has been changed to the International Federation for the Promotion of Mechanism and Machine Science.) It was named the Symposium on Theory and Practice of Robots and Manipulators (RoManSy). Its trademark was an emphasis on the mechanical sciences and the active participation of researchers from Eastern and Western Europe as well as North America and Japan. It is still held biannually. On a personal note, it is at RoManSy where I first met each of the editors of this Handbook: Dr. Khatib in 1978 and Dr. Siciliano in 1984. They were both students: Bruno Siciliano had been working on his PhD for about one year, and Oussama Khatib had just completed his PhD research. In both cases, it was love at first sight!

RoManSy was quickly joined by a host of other new conferences and workshops; today there are a large number of research oriented robotics meetings that take place through the year in many countries. Currently, the largest conference is the International Conference on Robotics and Automation (ICRA), which regularly draws well over 1000 participants.

In the beginning of the 1980s, the first real textbook on robotic manipulation in the USA was written by Richard *Lou* Paul (Richard P. Paul, *Robot Manipulators: Mathematics, Programming, and Control*, The MIT Press, Cambridge, MA, 1981). It used the idea of taking classical subjects in mechanics and applying them to robotics. In addition there were several topics developed directly from his thesis research at SAIL. (In the book, many examples are based on Scheinman's Stanford Arm.) Paul's book was a landmark event in the USA; it created a pattern for several influential future textbooks and also encouraged the creation of specialized robotics courses at a host of colleges and universities.

At about this same time, new journals were created to deal primarily with research papers in the areas related to robotics. The *International Journal of Robotics Research* was founded in the spring of 1982, and three years later the *IEEE Journal of Robotics and Automation* (now the *IEEE Transactions on Robotics*) was founded.

As microprocessors became ubiquitous, the question of what is or is not a robot came more into

play. This issue has, in my mind, never been successfully resolved. I do not think a definition will ever be universally agreed upon. There are of course the science fiction creatures-from-outer-space varieties, and the robots of the theater, literature, and the movies. There are examples of imaginary robot-like beings that predate the industrial revolution, but how about more down-to-Earth robots? In my view the definition is essentially a moving target that changes its character with technological progress. For example, when it was first developed, a ship's gyro auto-compass was considered a robot. Today, it is not generally included when we list the robots in our world. It has been demoted and is now considered an automatic control device.

For many, the idea of a robot includes the concept of multifunctionality, meaning the device is designed and built with the ability to be easily adapted or reprogrammed to do different tasks. In theory this idea is valid, but in practice it turns out that most robotic devices are multifunctional in only a very limited arena. In industry it was quickly discovered that a specialized machine, in general, performs much better than a general purpose machine. Furthermore, when the volume of production is high enough, a specialized machine can cost less to manufacture than a generalized one. So, specialized robots were developed for painting, riveting, quasiplanar parts assembly, press loading, circuit board stuffing, etc. In some cases robots are used in such specialized ways that it becomes difficult to draw the line between a so-called robot and an adjustable piece of *fixed* automation. Much of this practical unfolding is contrary to the dream of the pioneers in robotics, who had hoped for the development of general purpose machines that would do *everything*, and hence sell in great enough volume to be relatively inexpensive.

My view is that the notion of a robot has to do with which activities are, at a given time, associated with people and which are associated with machines. If a machine suddenly becomes able to do what we normally associate with people, the machine can be upgraded in classification and classified as a robot. After a while, people get used to the activity being done by machines, and the devices get downgraded from *robot* to *machine*. Machines that do not have fixed bases, and those that have arm- or leg-like appendages have the advantage of being more likely called robots, but it is hard to think of a consistent set of criteria that fits all the current naming conventions.

In actuality any machines, including familiar household appliances, which have microprocessors directing their actions can be considered as robots. In addition to vacuum cleaners, there are washing machines, refrigerators, and dishwashers that could be easily marketed as robotic devices. There are of course a wide range of possibilities, including those machines that have sensory environmental feedback and decision-making capabilities. In actual practice, in devices considered to be robotic, the amount of sensory and decision making capability may vary from a great deal to none.

In recent decades the study of robotics has expanded from a discipline centered on the study of mechatronic devices to a much broader interdisciplinary subject. An example of this is the area called human-centered robotics. Here one deals with the interactions between humans and intelligent machines. This is a growing area where the study of the interactions between robots and humans has enlisted expertise from outside the classical robotics domain. Concepts such as emotions in both robots and people are being studied, and older areas such as human physiology and biology are being incorporated into the mainstream of robotics research. These activities enrich the field of robotics, as they introduce new engineering and science dimensions into the research discourse.

Originally, the nascent robotics community was focused on getting things to work. Many early devices were remarkable in that they worked at all, and little notice was taken of their limited performance. Today, we have sophisticated, reliable devices as part of the modern array of robotic systems. This progress is the result of the work of thousands of people throughout the world. A lot of this work took place in universities, government research laboratories, and companies. It is a tribute to the worldwide engineering and scientific community that it has been able to create the vast amount of information that is contained in the 64 chapters of this Handbook. Clearly these results did not arise by any central planning or by an overall orderly scheme. So the editors of this handbook were faced with the difficult task of organizing the material into a logical and coherent whole.

The editors have accomplished this by organizing the contributions into a three-layer structure. The first layer deals with the *foundations* of the subject. This layer consists of a single part of nine chapters in which the authors lay out the root subjects: kinematics, dynamics, control, mechanisms, architecture, programming, reasoning, and sensing. These are the basic technological building blocks for robotics study and development.

The second layer has four parts. The first of these deals with *robot structures*; these are the arms, legs, hands, and other parts that most robots are made up of. At first blush, the hardware of legs, arms, and hands may look quite different from each other, yet they share a common set of attributes that allows them to all be treated with the same, or closely related, aspects of the fundamentals described in the first layer.

The second part of this layer deals with *sensing and perception*, which are basic abilities any truly autonomous robotic system must have. As was pointed out earlier, in practice, many so-called robotic devices have little of these abilities, but clearly the more advanced robots cannot exist without them, and the trend is very much toward incorporating such capabilities into robotic devices. The third part of this layer treats the subject areas associated with the technology of *manipulation and the interfacing of devices*. The fourth part of this layer is made up of eight chapters that treat *mobile robots and various forms of distributed robotics*.

The third layer consists of two separate parts (a total of 22 chapters) that deal with advanced applications at the forefront of today's research and development. There are two parts to this layer; one deals with *field and service robots*, and the other deals with *human-centered and lifelike robots*. To the uninitiated observer, these chapters are what advanced robotics is all about. However, it is important to realize that many of these extraordinary accomplishments would probably not exist without the previous developments introduced in the first two layers of this Handbook.

It is this intimate connection between theory and practice that has nurtured the growth of robotics and become a hallmark of modern robotics. These two complementary aspects have been a source of great personal satisfaction to those of us who have had the opportunity to both research and develop robotic devices. The contents of this Handbook admirably reflect this complementary aspect of the subject, and present a very useful bringing together of the vast accomplishments which have taken place in the last 50 years. Certainly, the contents of this Handbook will serve as a valuable tool and guide to those who will produce the even more capable and diverse next generations of robotic devices. The editors and authors have my congratulations and admiration.

Stanford, August 2007 Bernard Roth

Foreword

Georges Giralt
Emeritus Research Director
LAAS-CNRS Toulouse
(deceased)

To open this Handbook and unfold the richness of its 64 chapters, we here attempt a brief personal overview to sketch the evolution of robotics in its many aspects, concepts, trends, and central issues.

The modern story of Robotics began about half a century ago with developments in two different directions.

First, let us acknowledge the domain of mechanical arms, ranging from teleoperated tasks on radiation-contaminated products to industrial arms, with the landmark machine UNIMATE – standing for uni(versal)mate. The industrial development of products, mostly around the six-degree-of-freedom serial links paradigm and active research and development, associating mechanical engineering to the control specialism, was the main driving force here. Of particular note nowadays is the successfully pursued effort to design novel application-optimized structures, using powerful sophisticated mathematical tools. In a similar way, an important issue concerns the design and the actual building of arms and hands in the context of human-friendly robots for tomorrow's cognitive robot.

Second, and less well recognized, we should acknowledge the stream of work concerned with themes in artificial intelligence. A landmark project in this area was the mobile robot *Shakey* developed at Stanford International. This work, which aimed to bring together computer science, artificial intelligence, and applied mathematics to develop intelligent machines, remained a secondary area for quite some time. During the 1980s, building strength from many study cases encompassing a spectacular spectrum ranging from rovers for extreme environments (planet exploration, Antarctica, etc.), to service robots (hospitals, museum guides, etc.), a broad research domain arose in which machines could claim the status of intelligent robots.

Hence robotics researches could bring together these two different branches, with intelligent robots categorized in a solely computational way as bounded rationality machines, expanding on the 1980s third-generation robot definition:

> *(robot) . . . operating in the three-dimensional world as a machine endowed with the capacity to interpret and to reason about a task and about its execution, by intelligently relating perception to action.*

The field of autonomous robots, a widely recognized test-bed, has recently benefited from salient contributions in robot planning using the results of algorithmic geometry as well as of a stochastic framework approach applied both to environmental modeling and robot localization problems (SLAM, simultaneous localization and modeling), and further from the development of decisional procedures via Bayesian estimation and decision approaches.

For the last decade of the millennium, robotics largely dealt with the intelligent robot paradigm, blending together robots and machine-intelligence generic research within themes covering advanced sensing and perception, task reasoning and planning, operational and decisional autonomy, functional integration architectures, intelligent human–machine interfaces, safety, and dependability.

The second branch, for years referred to as non-manufacturing robotics, concerns a wide spectrum of research-driven real-world cases pertaining to field, service, assistive, and, later, personal robotics. Here, machine intelligence is, in its various themes, the central research direction, enabling the robot to act:

1. As a human surrogate, in particular for intervention tasks in remote and/or hostile environments
2. In close interaction with humans and operating in human environments in all applications encompassed by human-friendly robotics, also referred to as human-centered robotics
3. In tight synergy with the user, expanding from mechanical exoskeleton assistance, surgery, health care, and rehabilitation into human augmentation.

Consequently, at the turn of the millennium, robotics appears as a broad spectrum of research themes both supporting market products for well-engineered industrial workplaces, and a large number of domain-oriented application cases operating in hazardous and/or harsh environments (underwater robotics, rough-terrain rovers, health/rehabilitation care robotics, etc.) where robots exhibit meaningful levels of shared autonomy.

The evolution levels for robotics stress the role of theoretical aspects, moving from application domains to the technical and scientific area. The organization of this Handbook illustrates very well these different lev-

els. Furthermore, it rightly considers, besides a body of software systems, front-line matters on physical appearance and novel appendages, including legs, arms, and hands design in the context of human-friendly robots for tomorrow's cognitive robot.

Forefront robotics in the first decade of the current millennium is making outstanding progress, compounding the strength of two general directions:

- Short/mid-term application-oriented study cases
- Mid/long-term generic situated research.

For completeness, we should mention the large number of peripheral, robotics-inspired subjects, quite often concerning entertainment, advertising, and sophisticated toys.

The salient field of human-friendly robotics encompasses several front-line application domains where the robots operate in a human environment and in close interaction with humans (entertainment and education, public-oriented services, assistive and personal robots, etc.), which introduces the critical issue of human–robot interaction.

Right at the core of the field, emerges the forefront topic of personal robots for which three general characteristics should be emphasized:

1. They may be operated by a nonprofessional user;
2. They may be designed to share high-level decision making with the human user;
3. They may include a link to environment devices and machine appendages, remote systems, and operators; the shared decisional autonomy concept (co-autonomy) implied here unfolds into a large set of cutting-edge research issues and ethical problems.

The concept of the personal robot, expanding to robot assistant and universal companion, is a truly great challenge for robotics as a scientific and technical field, offering the mid/long-term perspective of achieving a paramount societal and economical impact. This introduces, and questions, front-line topics encompassing cognitive aspects: user-tunable human–machine intelligent interfaces, perception (scene analysis, category identification), open-ended learning (understanding the universe of action), skills acquisition, extensive robot-world data processing, decisional autonomy, and dependability (safety, reliability, communication, and operating robustness).

There is an obvious synergistic effort between the two aforementioned approaches, in spite of the necessary framework time differences. The scientific link not only brings together the problems and obtained results but also creates a synergistic exchange between the two sides and the benefits of technological progress.

Indeed, the corresponding research trends and application developments are supported by an explosive evolution of enabling technologies: computer processing power, telecommunications, networking, sensing devices, knowledge retrieval, new materials, micro- and nanotechnologies.

Today, looking to the mid- and long-term future, we are faced with very positive issues and perspectives but also having to respond to critical comments and looming dangers for machines that are in physical contact with the user and may also be capable of unwanted, unsafe behavior. Therefore, there is a clear need to include at the research level safety issues and the topic of multifaced dependability and the corresponding system constraints.

The *Handbook of Robotics* is an ambitious and timely endeavor. It summarizes a large number of problems, questions, and facets considered by 164 authors in 64 chapters. As such it not only provides an efficient display of basic topics and results obtained by researches around the world, but furthermore gives access to this variety of viewpoints and approaches to everyone. This is indeed an important tool for progress but, much more, is the central factor that will establish the two first decades of this millennium as the dawn of robotics, lifted to a scientific discipline at the core of machine intelligence.

Toulouse, December 2007 Georges Giralt

Foreword

The field of robotics was born in the middle of the last century when emerging computers were altering every field of science and engineering. Having gone through fast yet steady growth via a procession of stages from infancy, childhood, and adolescence to adulthood, robotics is now mature and is expected to enhance the quality of people's lives in society in the future.

In its infancy, the core of robotics consisted of pattern recognition, automatic control, and artificial intelligence. Taking on these new challenge, scientists and engineers in these fields gathered to investigate novel robotic sensors and actuators, planning and programming algorithms, and architectures to connect these components intelligently. In so doing, they created artifacts that could interact with humans in the real world. An integration of these early robotics studies yielded *hand–eye systems*, the test-bed of artificial intelligence research.

The playground for childhood robotics was the factory floor. Industrial robots were invented and introduced into the factory for automating spraying, spot welding, grinding, materials handling, and parts assembly. Machines with sensors and memories made the factory floor smarter, and its operations more flexible, reliable, and precise. Such robotic automation freed humans from heavy and tedious labor. The automobile, electric appliance, and semiconductor industries rapidly retooled their manufacturing lines into robot-integrated systems. In the late 1970s, the word *mechatronics*, originally coined by the Japanese, defined a new concept of machinery, one in which electronics was fused with mechanical systems, making a wide range of industrial products simpler, more functional, programmable, and intelligent. Robotics and mechatronics exerted an evolutionary impact on the design and operation of manufacturing processes as well as on manufactured products.

As robotics entered its adolescence, researchers were ambitious to explore new horizons. Kinematics, dynamics, and control system theory were refined and applied to real complex robot mechanisms. To plan and carry out real tasks, robots had to be made cognizant of their surroundings. Vision, the primary channel for external sensing, was exploited as the most general, effective, and efficient means for robots to understand their external situation. Advanced algorithms and powerful devices were developed to improve the speed and robustness of robot vision systems. Tactile and force sensing systems also needed to be developed for robots to manipulate objects. Studies on modeling, planning, knowledge, reasoning, and memorization expanded their intelligent properties. Robotics became defined as the study of intelligent connection of sensing to actuation. This definition covered all aspects of robotics: three scientific cores and one synthetic approach to integrate them. Indeed, system integration became a key aspect of robotic engineering as it allows the creation of lifelike machines. The fun of creating such robots attracted many students to the robotics field.

Hirochika Inoue
Professor Emeritus
The University of Tokyo

In advancing robotics further, scientific interest was directed at understanding humans. Comparative studies of humans and robots led to new approaches in scientific modeling of human functions. Cognitive robotics, lifelike behavior, biologically inspired robots, and a psychophysiological approach to robotic machines culminated in expanding the horizons of robotic potential. Generally speaking, an immature field is sparse in scientific understanding. Robotics in the 1980s and 1990s was in such a youthful stage, attracting a great many inquisitive researchers to this new frontier. Their continuous explorations into new realms form the rich scientific contents of this comprehensive volume.

Further challenges, along with expertise acquired on the cutting edge of robotics, opened the way to real-world applications for mature robotics. The early-stage playground gave way to a workshop for industrial robotics. Medical robotics, robot surgery, and in vivo imaging save patients from pain while providing doctors with powerful tools for conducting operations. New robots in such areas as rehabilitation, health care, and welfare are expected to improve quality of life in an aging society. It is the destiny of robots to go everywhere, in the air, under water, and into space. They are expected to work hand in hand with humans in such areas as agriculture, forestry, mining, construction, and hazardous environments and rescue operations, and to find utility both in domestic work and in providing services in shops, stores, restaurants, and hospitals. In a myriad of ways, robotic devices are expected to support our daily lives. At this point, however, robot applications are largely limited to structured environments, where they are separated from humans for safety sake.

In the next stage, their environment will be expanded to an unstructured world, one in which humans, as service takers, will always live and work beside robots. Improved sensing, more intelligence, enhanced safety, and better human understanding will be needed to prepare robots to function in such an environment. Not only technical but also social matters must be considered in finding solutions to issues impeding this progress.

Since my initial research to make a robot turn a crank, four decades have passed. I feel both lucky and happy to have witnessed the growth of robotics from its early beginnings. To give birth to robotics, fundamental technologies were imported from other disciplines. Neither textbooks nor handbooks were available. To reach the present stage, a great many scientists and engineers have challenged new frontiers; advancing robotics, they have enriched this body of knowledge from a variety of perspectives. The fruits of their endeavors are compiled in this *Handbook of Robotics*. More than 100 of the world's leading experts have collaborated in producing this publication. Now, people who wish to commit themselves to robotics research can find a firm foundation to build upon. This Handbook is sure to be used to further advance robotics science, reinforce engineering education, and systematically compile knowledge that will innovate both society and industry.

The roles of humans and robots in an aging society pose an important issue for scientists and engineers to consider. Can robotics contribute to securing peace, prosperity, and a greater quality of life? This is still an open question. However, recent advances in personal robots, robotic home appliances, and humanoids suggest a paradigm shift from the industrial to the service sector. To realize this, robotics must be addressed from such viewpoints as the working infrastructure within society, psychophysiology, law, economy, insurance, ethics, art, design, drama, and sports science. Future robotics should be studied as a subject that envelops both humanity and technology. This Handbook offers a selected technical foundation upon which to advance such newly emerging fields of robotics. I look forward to continuing progress adding page after page of robot-based prosperity to future society.

Tokyo, September 2007 Hirochika Inoue

Foreword

Robots have fascinated people for thousands of years. Those automatons that were built before the 20th century did not connect sensing to action but rather operated through human agency or as repetitive machines. However, by the 1920s electronics had gotten to the stage that the first true robots that sensed the world and acted in it appropriately could be built. By 1950 we started to see descriptions of real robots appearing in popular magazines. By the 1960s industrial robots came onto the scene. Commercial pressures made them less and less responsive to their environments but faster and faster in what they did in their carefully engineered world. Then in the mid 1970s in France, Japan, and the USA we started to see robots rising again in a handful of research laboratories, and now we have arrived at a world-wide frenzy in research and the beginnings of large-scale deployment of intelligent robots throughout our world. This Handbook brings together the current state of robotics research in one place. It ranges from the mechanism of robots through sensing and perceptual processing, intelligence, action, and many application areas.

I have been more than fortunate to have lived with this revolution in robotics research over the last 30 years. As a teenager in Australia I built robots inspired by the tortoises of Walter described in the *Scientific American* in 1949 and 1950. When I arrived in Silicon Valley in 1977, just as the revolution in the personalization of computation was really coming into being, I instead turned to the much more obscure world of robots. In 1979 I was able to assist Hans Moravec at the Stanford Artificial Intelligence Lab (SAIL) as he coaxed his robot *The Cart* to navigate 20 m in 6 hours. Just 26 years later, in 2005, at the same laboratory, SAIL, Sebastian Thrun and his team coaxed their robot to autonomously drive 200 000 m in 6 hours: four orders of magnitude improvement in a mere 26 years, which is slightly better than a doubling every 2 years. However, robots have not just improved in speed, they have also increased in number. When I arrived at SAIL in 1977 we knew of three mobile robots operating in the world. Recently a company that I founded manufactured its 3 000 000th mobile robot, and the pace is increasing. Other aspects of robots have had similarly spectacular advances, although it is harder to provide such crisp numeric characterizations. In recent years we have gone from robots being too unaware of their surroundings that it was unsafe for people to share their workspace to robots that people can work with in close contact, and from robots that were totally unaware of people to robots that pick up on natural social cues from facial expressions to prosody in people's voices. Recently robotics has crossed the divide between flesh and machines so that now we are seeing neurorobotics ranging from prosthetic robotic extensions to rehabilitative robots for the disabled. And very recently robotics has become a respected contributor to research in cognitive science and neuroscience.

Rodney Brooks
Panasonic Professor of Robotics
Massachusetts Institute of Technology

The research results chronicled in this volume give the key ideas that have enabled these spectacular advances. The editors, the part editors, and all the contributors have done a stellar job in bring this knowledge together in one place. Their efforts have produced a work that will provide a basis for much further research and development. Thank you, and congratulations to all who have labored on this pivotal book.

Some of the future robotics research will be incremental in nature, taking the state of the art and improving upon it. Other parts of future research will be more revolutionary, based on ideas that are antithetical to some of the ideas and current state of the art presented in this book.

As you study this volume and look for places to contribute to research through your own talents and hard work I want to alert you to capabilities or aspirations that I believe will make robots even more useful, more productive, and more accepted. I describe these capabilities in terms of the age at which a child has equivalent capabilities:

- The object-recognition capabilities of a 2-year-old child
- The language capabilities of a 4-year-old child
- The manual dexterity of a 6-year-old child
- The social understanding of an 8-year-old child.

Each of these is a very difficult goal. However even small amounts of progress towards any one of these goals will have immediate applications to robots out in the world. Good reading and best wishes as you contribute further to robotkind.

Cambridge, October 2007 Rodney Brooks

Preface to the Second Edition

The Springer Handbook of Robotics was a challenging six-year endeavour from 2002 to 2008. It mobilized a large number of active scientists and researchers to produce this unique comprehensive reference source combining basic and advanced developments. The handbook has been very successful and extremely well received in our community. New researchers have been attracted to robotics which in turn have contributed to further progress in this trans-disciplinary field.

The handbook soon established itself as a landmark in robotics publishing and beyond. It has been the bestseller of all Springer engineering books during the last seven years, the number one in chapter downloads (nearly forty thousand a year), and the fourth most downloaded over all Springer books in 2011. In February 2009, the handbook was recognized as the Winner of the American Association of Publishers (AAP) PROSE Award for Excellence in Physical Sciences & Mathematics as well as the Award for Engineering & Technology.

The rapid growth of our field as well as the birth of new research areas motivated us in 2011 to start pursuing a second edition with the intent to provide not only an update but also an expansion of the handbook's contents. Our editorial board (with David Orin, Frank Park, Henrik Christensen, Makoto Kaneko, Raja Chatila, Alex Zelinsky, and Daniela Rus) has been enthusiastically engaged during the last four years to coordinate the contributions of the authors to the seven parts of the handbook in its three-layer structure. The contents have been restructured to achieve four main objectives: the enlargement of foundational topics for robotics, the enlightenment of design of various types of robotic systems, the extension of the treatment on robots moving in the environment, and the enrichment of advanced robotics applications. Most previous chapters have been revised, fifteen new chapters have been introduced on emerging topics, and a new generation of authors have joined the handbook's team. The contents were finalized by the spring of 2015 after extensive review and feedback, and the project was completed by the fall of 2015 – generating, by that time, a record of over 12 000 additional emails in our folders to the 10 000 of the first edition. The result is an impressive collection of 80 chapters over the 7 parts, contributed by 229 authors, with more than 2300 pages, 1375 illustrations and 9411 references.

One of the major additions of the second edition of the handbook is the inclusion of multimedia material. An editorial team has been established under the leadership of Torsten Kröger and the contributions of Gianluca Antonelli, Dongjun Lee, Dezhen Song and Stefano Stramigioli. With the commitment of such a group of energetic young scholars, the multimedia project has been pursued in parallel to the handbook project. The multimedia editorial team has selected for each chapter video contributions, from those suggested by the authors, based on their quality and relevance to the chapter's contents. In addition, the handbook editors have produced tutorial videos that can be accessed directly from each part of the handbook. An openly accessible multimedia website, http://handbookofrobotics.org, has been established to host these videos with the sponsorship of IEEE Robotics and Automation Society and Google. The website has been conceived as a live dissemination project bringing the latest robotics contributions to the world community.

We are deeply grateful for the continuous commitment of our handbook extended team, particularly the newcomers to the project. We would like to express our gratitude and appreciation to Judith Hinterberg, Werner Skolaut and Thomas Ditzinger from Springer for their strong support, as well as to Anne Strohbach and the le-tex staff for their highly professional typesetting work in the production.

Eight years after the first appearance of the handbook, the second edition comes to light. Beyond its tutorial value for our community, it is our conviction that the handbook will continue to serve as a useful source to attract new researchers to robotics and inspire decades of vibrant progress in this fascinating field. The cooperative spirit inspiring our team since the inception of the first edition is amusingly illustrated in the video *The Handbook – A Short History* (VIDEO 844). The completion of the second edition has been inspired by that same spirit and the gradient has been kept :-) Our fellows in the robotics community are reminded now to … keep the Hessian ;-)

January 2016

Bruno Siciliano — Naples

Oussama Khatib — Stanford

Preface to the Multimedia Extension

Scientific and technical advancements in the domain of robotics have accelerated significantly over the past decade. Since the inception of the Second Edition of the Springer Handbook of Robotics in 2011, the Editors Bruno Siciliano and Oussama Khatib decided to add multimedia content and appointed an editorial team: Gianluca Antonelli, Dongjun Lee, Dezhen Song, Stefano Stramigioli, and myself as the Multimedia Editor.

Over the five years of the project, everyone on the team worked with all of the 229 authors, the Part Editors, and the Editors. Besides communicating with all 80 Authors' teams and reviewing, selecting, and improving all video contributions, we also scanned all the videos published at robotics conferences organized by the IEEE Robotics and Automation Society since 1991. A total of more than 5500 e-mails were sent back and forth to coordinate the project and to ensure the quality of the content. We implemented a video management system that allows authors to upload videos, editors to review videos, and readers to access videos. Videos were selected with the goal of helping convey content to all readers of the Second Edition. They may be relevant from a technical, scientific, educational, or historical perspective. All chapter and part videos are publicly accessible and can be found at

http://handbookofrobotics.org

In addition to the videos referenced in the chapters, each of the seven parts is accompanied by a part video giving an overview of each part. The storyboards of these videos were created by the Part Editors and then professionally produced.

The video content provided in the Multimedia Extension makes understanding the written content easier and was designed to be a comprehensive addition to the Handbook. Concepts, methods, experiments, and applications described in the book were animated, visually illustrated, or paired with sound and narration – giving readers a further dimension to comprehend the written content of the book.

Coordinating the work with more than 200 contributors cannot just be done by a small team, and we are deeply grateful for the support of many people and organizations. Judith Hinterberg and Thomas Ditzinger from the Springer Team in Heidelberg helped us tremendously with professional support during the entire production phase. The app for smartphones and tablets was implemented by Rob Baldwin from Studio Orb and allows readers easy access to multimedia content. The IEEE Robotics and Automation Society granted permissions to use all videos that have been published in the proceedings of conferences sponsored by the society. Google and X supported us by donating funds for the implementation of the website backend.

Following the Editors' inspiration, let us keep working and communicating as one community – and let us keep the Hessian all together . . . !

March 2016
Torsten Kröger Mountain View

Accessing Multimedia Contents

Multimedia contents are an integral part of the Second Edition of the Springer Handbook of Robotics. 69 chapters contain video icons like this one:



Each icon indicates a video ID that can be used to access individual videos in various simple and intuitive ways.

Using the Multimedia App

We recommend using the multimedia app for smartphone and tablet PCs. You can install the app on *iOS* and *Android* devices using the QR code below. The app allows you to simply scan the pages of the book and automatically play all videos on your device while reading the book.

Multimedia Contents

Using the Website: http://handbookofrobotics.org

All chapter videos and part videos can be accessed directly from the website of the multimedia extension. Just enter a video ID in the search field in the top right corner of the website. You may also use the website to browse through chapter and part videos.

Using PDF Files

If you read an electronic copy of the Handbook, each video icon contains a hyper link. Just click on the link to watch the corresponding video.

Using QR Codes

Each chapter starts with a QR code that contains a link to all videos of the chapter. Part videos can be accessed through the QR code at the beginning of each part.

Contents

List of Abbreviations LXIII

1 **Robotics and the Handbook**
Bruno Siciliano, Oussama Khatib 1
1.1 A Brief History of Robotics 1
1.2 The Robotics Community 3
1.3 This Handbook 4
Video-References 5

Part A Robotics Foundations

2 **Kinematics**
Kenneth J. Waldron, James Schmiedeler 11
2.1 Overview 12
2.2 Position and Orientation Representation 12
2.3 Joint Kinematics 21
2.4 Geometric Representation 25
2.5 Workspace 27
2.6 Forward Kinematics 28
2.7 Inverse Kinematics 29
2.8 Forward Instantaneous Kinematics 31
2.9 Inverse Instantaneous Kinematics 32
2.10 Static Wrench Transmission 33
2.11 Conclusions and Further Reading 33
References 33

3 **Dynamics**
Roy Featherstone, David E. Orin 37
3.1 Overview 38
3.2 Spatial Vector Notation 39
3.3 Canonical Equations 45
3.4 Dynamic Models of Rigid-Body Systems 47
3.5 Kinematic Trees 51
3.6 Kinematic Loops 58
3.7 Conclusions and Further Reading 61
References 63

4 **Mechanism and Actuation**
Victor Scheinman, J. Michael McCarthy, Jae-Bok Song 67
4.1 Overview 68
4.2 System Features 68
4.3 Kinematics and Kinetics 69
4.4 Serial Robots 72
4.5 Parallel Robots 73
4.6 Mechanical Structure 75
4.7 Joint Mechanisms 76

4.8 Actuators 78
4.9 Robot Performance 85
4.10 Conclusions and Further Reading 87
Video-References 87
References 87

5 **Sensing and Estimation**
Henrik I. Christensen, Gregory D. Hager 91
5.1 Introduction 91
5.2 The Perception Process 92
5.3 Sensors 94
5.4 Estimation Processes 98
5.5 Representations 109
5.6 Conclusions and Further Readings 111
References 111

6 **Model Identification**
John Hollerbach, Wisama Khalil, Maxime Gautier 113
6.1 Overview 113
6.2 Kinematic Calibration 115
6.3 Inertial Parameter Estimation 122
6.4 Identifiability and Numerical Conditioning 127
6.5 Conclusions and Further Reading 135
Video-References 136
References 137

7 **Motion Planning**
Lydia E. Kavraki, Steven M. LaValle 139
7.1 Robotics Motion Planning 139
7.2 Motion Planning Concepts 140
7.3 Sampling-Based Planning 141
7.4 Alternative Approaches 144
7.5 Differential Constraints 148
7.6 Extensions and Variations 151
7.7 Advanced Issues 154
7.8 Conclusions and Further Reading 157
Video-References 158
References 158

8 **Motion Control**
Wan Kyun Chung, Li-Chen Fu, Torsten Kröger 163
8.1 Introduction to Motion Control 164
8.2 Joint Space Versus Operational Space Control 166
8.3 Independent-Joint Control 167
8.4 PID Control 169
8.5 Tracking Control 172
8.6 Computed-Torque Control 174
8.7 Adaptive Control 177
8.8 Optimal and Robust Control 181
8.9 Trajectory Generation and Planning 183
8.10 Digital Implementation 187

8.11 Learning Control 190
Video-References 191
References 191

9 **Force Control**
Luigi Villani, Joris De Schutter 195
9.1 Background 195
9.2 Indirect Force Control 198
9.3 Interaction Tasks 205
9.4 Hybrid Force/Motion Control 211
9.5 Conclusions and Further Reading 216
Video-References 217
References 218

10 **Redundant Robots**
Stefano Chiaverini, Giuseppe Oriolo, Anthony A. Maciejewski 221
10.1 Overview 221
10.2 Task-Oriented Kinematics 224
10.3 Inverse Differential Kinematics 227
10.4 Redundancy Resolution via Optimization 232
10.5 Redundancy Resolution via Task Augmentation 233
10.6 Second-Order Redundancy Resolution 236
10.7 Cyclicity 237
10.8 Fault Tolerance 237
10.9 Conclusion and Further Reading 239
Video-References 239
References 240

11 **Robots with Flexible Elements**
Alessandro De Luca, Wayne J. Book 243
11.1 Robots with Flexible Joints 244
11.2 Robots with Flexible Links 263
Video-References 279
References 279

12 **Robotic Systems Architectures and Programming**
David Kortenkamp, Reid Simmons, Davide Brugali 283
12.1 Overview 283
12.2 History 285
12.3 Architectural Components 289
12.4 Case Study – GRACE 296
12.5 The Art of Robot Architectures 298
12.6 Implementing Robotic Systems Architectures 299
12.7 Conclusions and Further Reading 302
Video-References 302
References 302

13 **Behavior-Based Systems**
François Michaud, Monica Nicolescu 307
13.1 Robot Control Approaches 308
13.2 Basic Principles of Behavior-Based Systems 310

13.3 Basis Behaviors 313
13.4 Representation in Behavior-Based Systems 313
13.5 Learning in Behavior-Based Systems 314
13.6 Applications and Continuing Work 318
13.7 Conclusions and Further Reading 322
Video-References 322
References 323

14 **AI Reasoning Methods for Robotics**
Michael Beetz, Raja Chatila, Joachim Hertzberg, Federico Pecora 329
14.1 Why Should a Robot Use AI-Type Reasoning? 330
14.2 Knowledge Representation and Processing 330
14.3 Reasoning and Decision Making 338
14.4 Plan-Based Robot Control 346
14.5 Conclusions and Further Reading 351
Video-References 351
References 352

15 **Robot Learning**
Jan Peters, Daniel D. Lee, Jens Kober, Duy Nguyen-Tuong, J. Andrew Bagnell, Stefan Schaal 357
15.1 What Is Robot Learning 358
15.2 Model Learning 360
15.3 Reinforcement Learning 372
15.4 Conclusions 385
Video-References 386
References 386

Part B Design

16 **Design and Performance Evaluation**
Jorge Angeles, Frank C. Park 399
16.1 The Robot Design Process 400
16.2 Workspace Criteria 401
16.3 Dexterity Indices 405
16.4 Other Performance Indices 408
16.5 Other Robot Types 411
16.6 Summary 416
References 416

17 **Limbed Systems**
Shuuji Kajita, Christian Ott 419
17.1 Design of Limbed Systems 420
17.2 Conceptual Design 420
17.3 Whole Design Process Example 423
17.4 Model Induced Design 427
17.5 Various Limbed Systems 434
17.6 Performance Indices 437
Video-References 439
References 440

18 **Parallel Mechanisms**
Jean-Pierre Merlet, Clément Gosselin, Tian Huang 443
18.1 Definitions 443
18.2 Type Synthesis of Parallel Mechanisms 445
18.3 Kinematics 446
18.4 Velocity and Accuracy Analysis 447
18.5 Singularity Analysis 448
18.6 Workspace Analysis 450
18.7 Static Analysis 451
18.8 Dynamic Analysis 452
18.9 Design 452
18.10 Wire-Driven Parallel Robots 453
18.11 Application Examples 455
18.12 Conclusion and Further Reading 455
Video-References 456
References 456

19 **Robot Hands**
Claudio Melchiorri, Makoto Kaneko 463
19.1 Basic Concepts 464
19.2 Design of Robot Hands 465
19.3 Technologies for Actuation and Sensing 470
19.4 Modeling and Control of a Robot Hand 473
19.5 Applications and Trends 477
19.6 Conclusions and Further Reading 478
Video-References 478
References 479

20 **Snake-Like and Continuum Robots**
Ian D. Walker, Howie Choset, Gregory S. Chirikjian 481
20.1 Snake Robots – Short History 481
20.2 Continuum Robots – Short History 485
20.3 Snake-Like and Continuum Robot Modeling 487
20.4 Modeling of Locomotion for Snake-Like and Continuum Mechanisms 491
20.5 Conclusion and Extensions to Related Areas 492
Video-References 492
References 493

21 **Actuators for Soft Robotics**
Alin Albu-Schäffer, Antonio Bicchi 499
21.1 Background 500
21.2 Soft Robot Design 502
21.3 Modeling Actuators for Soft Robotics 508
21.4 Modeling Soft Robots 511
21.5 Stiffness Estimation 513
21.6 Cartesian Stiffness Control 515
21.7 Periodic Motion Control 518
21.8 Optimal Control of Soft Robots 521
21.9 Conclusions and Open Problems 524

Video-References 525
References 526

22 **Modular Robots**
I-Ming Chen, Mark Yim 531
22.1 Concepts and Definitions 531
22.2 Reconfigurable Modular Manipulators 533
22.3 Self-Reconfigurable Modular Robots 535
22.4 Conclusion and Further Reading 539
Video-References 540
References 540

23 **Biomimetic Robots**
Kyu-Jin Cho, Robert Wood 543
23.1 Overview 544
23.2 Components of Biomimetic Robot Design 544
23.3 Mechanisms 545
23.4 Material and Fabrication 561
23.5 Conclusion 567
Video-References 568
References 570

24 **Wheeled Robots**
Woojin Chung, Karl Iagnemma 575
24.1 Overview 575
24.2 Mobility of Wheeled Robots 576
24.3 Wheeled Robot Structures 582
24.4 Wheel–Terrain Interaction Models 586
24.5 Wheeled Robot Suspensions 589
24.6 Conclusions 592
Video-References 592
References 593

25 **Underwater Robots**
Hyun-Taek Choi, Junku Yuh 595
25.1 Background 595
25.2 Mechanical Systems 596
25.3 Power Systems 599
25.4 Underwater Actuators and Sensors 601
25.5 Computers, Communications, and Architecture 606
25.6 Underwater Manipulators 614
25.7 Conclusions and Further Reading 617
Video-References 617
References 618

26 **Flying Robots**
Stefan Leutenegger, Christoph Hürzeler, Amanda K. Stowers, Kostas Alexis, Markus W. Achtelik, David Lentink, Paul Y. Oh, Roland Siegwart 623
26.1 Background and History 624
26.2 Characteristics of Aerial Robotics 625
26.3 Basics of Aerodynamics and Flight Mechanics 629
26.4 Airplane Modeling and Design 641

26.5 Rotorcraft Modeling and Design 647
26.6 Flapping Wing Modeling and Design 653
26.7 System Integration and Realization 659
26.8 Applications of Aerial Robots 662
26.9 Conclusions and Further Reading 666
Video-References 666
References 667

27 **Micro-/Nanorobots**
Bradley J. Nelson, Lixin Dong, Fumihito Arai 671
27.1 Overview of Micro- and Nanorobotics 671
27.2 Scaling 674
27.3 Actuation at the Micro- and Nanoscales 675
27.4 Imaging at the Micro- and Nanoscales 676
27.5 Fabrication 678
27.6 Microassembly 681
27.7 Microrobotics 687
27.8 Nanorobotics 692
27.9 Conclusions 704
Video-References 704
References 705

Part C Sensing and Perception

28 **Force and Tactile Sensing**
Mark R. Cutkosky, William Provancher 717
28.1 Overview 717
28.2 Sensor Types 718
28.3 Tactile Information Processing 725
28.4 Integration Challenges 730
28.5 Conclusions and Future Developments 731
Video-References 731
References 731

29 **Inertial Sensing, GPS and Odometry**
Gregory Dudek, Michael Jenkin 737
29.1 Odometry 737
29.2 Gyroscopic Systems 739
29.3 Accelerometers 742
29.4 IMU Packages 743
29.5 Satellite-Based Positioning (GPS and GNSS) 744
29.6 GPS-IMU Integration 749
29.7 Further Reading 750
29.8 Currently Available Hardware 750
References 751

30 **Sonar Sensing**
Lindsay Kleeman, Roman Kuc 753
30.1 Sonar Principles 754
30.2 Sonar Beam Pattern 756
30.3 Speed of Sound 758

30.4 Waveforms 758
30.5 Transducer Technologies 759
30.6 Reflecting Object Models 760
30.7 Artifacts 761
30.8 TOF Ranging 762
30.9 Echo Waveform Coding 765
30.10 Echo Waveform Processing 767
30.11 CTFM Sonar 769
30.12 Multipulse Sonar 772
30.13 Sonar Rings and Arrays 773
30.14 Motion Effects 775
30.15 Biomimetic Sonars 778
30.16 Conclusions 779
Video-References 780
References 780

31 **Range Sensing**
Kurt Konolige, Andreas Nüchter 783
31.1 Range Sensing Basics 783
31.2 Sensor Technologies 785
31.3 Registration 794
31.4 Navigation and Terrain Classification and Mapping 804
31.5 Conclusions and Further Reading 807
References 807

32 **3-D Vision for Navigation and Grasping**
Danica Kragic, Kostas Daniilidis 811
32.1 Geometric Vision 812
32.2 3-D Vision for Grasping 820
32.3 Conclusion and Further Reading 822
Video-References 822
References 822

33 **Visual Object Class Recognition**
Michael Stark, Bernt Schiele, Aleš Leonardis 825
33.1 Object Classes 825
33.2 Review of the State of the Art 826
33.3 Discussion and Conclusions 837
References 838

34 **Visual Servoing**
François Chaumette, Seth Hutchinson, Peter Corke 841
34.1 The Basic Components of Visual Servoing 842
34.2 Image-Based Visual Servo 843
34.3 Pose-Based Visual Servo 851
34.4 Advanced Approaches 854
34.5 Performance Optimization and Planning 856
34.6 Estimation of 3-D Parameters 858
34.7 Determining s^* and Matching Issues 859
34.8 Target Tracking 859

34.9 Eye-in-Hand and Eye-to-Hand Systems Controlled in the Joint Space 860
34.10 Under Actuated Robots 861
34.11 Applications 863
34.12 Conclusions 863
Video-References 863
References 863

35 **Multisensor Data Fusion**
Hugh Durrant-Whyte, Thomas C. Henderson 867
35.1 Multisensor Data Fusion Methods 867
35.2 Multisensor Fusion Architectures 880
35.3 Applications 885
35.4 Conclusions 889
Video-References 889
References 890

Part D Manipulation and Interfaces

36 **Motion for Manipulation Tasks**
James Kuffner, Jing Xiao 897
36.1 Overview 898
36.2 Task-Level Control 900
36.3 Manipulation Planning 904
36.4 Assembly Motion 911
36.5 Unifying Feedback Control and Planning 918
36.6 Conclusions and Further Reading 920
Video-References 923
References 923

37 **Contact Modeling and Manipulation**
Imin Kao, Kevin M. Lynch, Joel W. Burdick 931
37.1 Overview 931
37.2 Kinematics of Rigid-Body Contact 932
37.3 Forces and Friction 936
37.4 Rigid-Body Mechanics with Friction 939
37.5 Pushing Manipulation 942
37.6 Contact Interfaces and Modeling 943
37.7 Friction Limit Surface 946
37.8 Contacts in Grasping and Fixture Designs 949
37.9 Conclusions and Further Reading 950
Video-References 951
References 951

38 **Grasping**
Domenico Prattichizzo, Jeffrey C. Trinkle 955
38.1 Models and Definitions 956
38.2 Controllable Twists and Wrenches 961
38.3 Compliant Grasps 965
38.4 Restraint Analysis 967

38.5 Examples 975
38.6 Conclusion and Further Reading 985
Video-References 986
References 986

39 **Cooperative Manipulation**
Fabrizio Caccavale, Masaru Uchiyama 989
39.1 Historical Overview 990
39.2 Kinematics and Statics 991
39.3 Cooperative Task Space 995
39.4 Dynamics and Load Distribution 996
39.5 Task-Space Analysis 998
39.6 Control 999
39.7 Conclusions and Further Reading 1003
Video-References 1004
References 1004

40 **Mobility and Manipulation**
Oliver Brock, Jaeheung Park, Marc Toussaint 1007
40.1 Grasping and Manipulation 1009
40.2 Control 1013
40.3 Motion Generation 1017
40.4 Learning 1021
40.5 Perception 1025
40.6 Conclusions and Further Reading 1029
Video-References 1029
References 1030

41 **Active Manipulation for Perception**
Anna Petrovskaya, Kaijen Hsiao 1037
41.1 Perception via Manipulation 1037
41.2 Object Localization 1038
41.3 Learning About an Object 1049
41.4 Recognition 1054
41.5 Conclusions 1057
Video-References 1058
References 1058

42 **Haptics**
Blake Hannaford, Allison M. Okamura 1063
42.1 Overview 1064
42.2 Haptic Device Design 1068
42.3 Haptic Rendering 1071
42.4 Control and Stability of Force Feedback Interfaces 1073
42.5 Other Types of Haptic Interfaces 1075
42.6 Conclusions and Further Reading 1079
References 1079

43 **Telerobotics**
Günter Niemeyer, Carsten Preusche, Stefano Stramigioli, Dongjun Lee 1085
43.1 Overview and Terminology 1085

43.2 Telerobotic Systems and Applications 1087
43.3 Control Architectures 1090
43.4 Bilateral Control and Force Feedback 1095
43.5 Emerging Applications of Telerobotics 1101
43.6 Conclusions and Further Reading 1104
Video-References 1104
References 1105

44 **Networked Robots**
Dezhen Song, Ken Goldberg, Nak-Young Chong 1109
44.1 Overview and Background 1109
44.2 A Brief History 1110
44.3 Communications and Networking 1112
44.4 Properties of Networked Robots 1115
44.5 Cloud Robotics 1121
44.6 Conclusion and Future Directions 1125
Video-References 1126
References 1126

Part E Moving in the Environment

45 **World Modeling**
Wolfram Burgard, Martial Hebert, Maren Bennewitz 1135
45.1 Historical Overview 1136
45.2 Models for Indoors and Structured Environments 1137
45.3 World and Terrain Models for Natural Environments 1141
45.4 Dynamic Environments 1149
45.5 Summary and Further Reading 1149
Video-References 1150
References 1150

46 **Simultaneous Localization and Mapping**
Cyrill Stachniss, John J. Leonard, Sebastian Thrun 1153
46.1 SLAM: Problem Definition 1154
46.2 The Three Main SLAM Paradigms 1157
46.3 Visual and RGB-D SLAM 1166
46.4 Conclusion and Future Challenges 1169
Video-References 1170
References 1171

47 **Motion Planning and Obstacle Avoidance**
Javier Minguez, Florant Lamiraux, Jean-Paul Laumond 1177
47.1 Nonholonomic Mobile Robots: Where Motion Planning Meets Control Theory 1178
47.2 Kinematic Constraints and Controllability 1179
47.3 Motion Planning and Small-Time Controllability 1180
47.4 Local Steering Methods and Small-Time Controllability 1181
47.5 Robots and Trailers 1184
47.6 Approximate Methods 1186
47.7 From Motion Planning to Obstacle Avoidance 1187

47.8 Definition of Obstacle Avoidance 1187
47.9 Obstacle Avoidance Techniques 1188
47.10 Robot Shape, Kinematics, and Dynamics in Obstacle Avoidance 1194
47.11 Integration Planning – Reaction 1196
47.12 Conclusions, Future Directions, and Further Reading 1198
Video-References 1199
References 1199

48 **Modeling and Control of Legged Robots**
Pierre-Brice Wieber, Russ Tedrake, Scott Kuindersma 1203
48.1 A Brief History of Legged Robots 1204
48.2 The Dynamics of Legged Locomotion 1204
48.3 Stability Analysis – Not Falling Down 1209
48.4 Generation of Dynamic Walking and Running Motions 1214
48.5 Motion and Force Control 1222
48.6 Towards More Efficient Walking 1225
48.7 Different Contact Behaviors 1227
48.8 Conclusion 1228
References 1228

49 **Modeling and Control of Wheeled Mobile Robots**
Claude Samson, Pascal Morin, Roland Lenain 1235
49.1 Background 1236
49.2 Control Models 1238
49.3 Adaptation of Control Methods for Holonomic Systems 1240
49.4 Methods Specific to Nonholonomic Systems 1241
49.5 Path Following in the Case of Nonideal Wheel-Ground Contact 1255
49.6 Complementary Issues and Bibliographical Guide 1261
Video-References 1263
References 1263

50 **Modeling and Control of Robots on Rough Terrain**
Keiji Nagatani, Genya Ishigami, Yoshito Okada 1267
50.1 Overview 1268
50.2 Modeling of Wheeled Robot in Rough Terrain 1270
50.3 Control of Wheeled Robot in Rough Terrain 1274
50.4 Modeling of Tracked Vehicle on Rough Terrain 1276
50.5 Stability Analysis of Tracked Vehicles 1278
50.6 Control of Tracked Vehicle on Rough Terrain 1279
50.7 Summary 1281
Video-References 1281
References 1282

51 **Modeling and Control of Underwater Robots**
Gianluca Antonelli, Thor I. Fossen, Dana R. Yoerger 1285
51.1 The Expanding Role of Marine Robotics in Oceanic Engineering 1285
51.2 Underwater Robotics 1287
51.3 Applications 1302
51.4 Conclusions and Further Reading 1303
Video-References 1304
References 1304

52 **Modeling and Control of Aerial Robots**
Robert Mahony, Randal W. Beard, Vijay Kumar 1307
52.1 Overview 1307
52.2 Modeling Aerial Robotic Vehicles 1309
52.3 Control 1316
52.4 Trajectory Planning 1324
52.5 Estimating the Vehicle State 1328
52.6 Conclusion 1330
Video-References 1331
References 1331

53 **Multiple Mobile Robot Systems**
Lynne E. Parker, Daniela Rus, Gaurav S. Sukhatme 1335
53.1 History 1336
53.2 Architectures for Multirobot Systems 1337
53.3 Communication 1339
53.4 Networked Mobile Robots 1340
53.5 Swarm Robots 1351
53.6 Modular Robotics 1354
53.7 Heterogeneity 1357
53.8 Task Allocation 1359
53.9 Learning 1361
53.10 Applications 1362
53.11 Conclusions and Further Reading 1366
Video-References 1366
References 1367

Part F Robots at Work

54 **Industrial Robotics**
Martin Hägele, Klas Nilsson, J. Norberto Pires, Rainer Bischoff 1385
54.1 Industrial Robotics: The Main Driver for Robotics Research and Application 1386
54.2 A Short History of Industrial Robots 1386
54.3 Industrial Robot Kinematics 1392
54.4 Typical Industrial Robot Applications 1393
54.5 Safe Human–Robot Collaboration 1405
54.6 Task Descriptions – Teaching and Programming 1409
54.7 System Integration 1414
54.8 Outlook and Long-Term Challenges 1416
Video-References 1418
References 1418

55 **Space Robotics**
Kazuya Yoshida, Brian Wilcox, Gerd Hirzinger, Roberto Lampariello 1423
55.1 Historical Developments and Advances of Orbital Robotic Systems . 1424
55.2 Historical Developments and Advances of Surface Robotic Systems. 1430
55.3 Mathematical Modeling 1437
55.4 Future Directions of Orbital and Surface Robotic Systems 1452
55.5 Conclusions and Further Reading 1457

Video-References 1457
References 1458

56 **Robotics in Agriculture and Forestry**
Marcel Bergerman, John Billingsley, John Reid, Eldert van Henten 1463
56.1 Section Scope 1464
56.2 Challenges and Opportunities 1465
56.3 Case Studies 1467
56.4 Conclusion 1487
Video-References 1488
References 1489

57 **Robotics in Construction**
Kamel S. Saidi, Thomas Bock, Christos Georgoulas 1493
57.1 Overview 1494
57.2 Offsite Applications of Robotics in Construction 1499
57.3 Onsite Applications of Single Task Construction Robots 1504
57.4 Integrated Robotized Construction Sites 1511
57.5 Currently Unsolved Technical Problems 1514
57.6 Future Directions 1516
57.7 Conclusions and Further Reading 1516
Video-References 1517
References 1517

58 **Robotics in Hazardous Applications**
James Trevelyan, William R. Hamel, Sung-Chul Kang 1521
58.1 Operation in Hazardous Environments: The Need for a Robotics Solution 1521
58.2 Applications 1523
58.3 Enabling Technologies 1537
58.4 Conclusions and Further Reading 1544
Video-References 1545
References 1546

59 **Robotics in Mining**
Joshua A. Marshall, Adrian Bonchis, Eduardo Nebot, Steven Scheding 1549
59.1 Modern Mining Practice 1550
59.2 Surface Mining 1552
59.3 Underground Mining 1562
59.4 Challenges and Industry Acceptance 1568
59.5 Challenges, Outlook, and Conclusion 1569
Video-References 1571
References 1572

60 **Disaster Robotics**
Robin R. Murphy, Satoshi Tadokoro, Alexander Kleiner 1577
60.1 Overview 1578
60.2 Disaster Characteristics and Impact on Robots 1581
60.3 Robots Actually Used at Disasters 1582
60.4 Robots at the Fukushima-Daiichi Nuclear Power Plant Accident 1588
60.5 Lessons Learned, Challenges, and Novel Approaches 1591

60.6 Evaluation 1598
60.7 Conclusions and Further Reading 1600
Video-References 1601
References 1601

61 **Robot Surveillance and Security**
Wendell H. Chun, Nikolaos Papanikolopoulos 1605
61.1 Overview 1605
61.2 Application Domains 1607
61.3 Enabling Technologies 1608
61.4 Active Research 1617
61.5 Conclusion 1622
Video-References 1623
References 1623

62 **Intelligent Vehicles**
Alberto Broggi, Alex Zelinsky, Ümit Özgüner, Christian Laugier 1627
62.1 The Motivation and Approaches to Intelligent Vehicles 1628
62.2 Enabling Technologies 1632
62.3 Road Scene Understanding 1635
62.4 Advanced Driver Assistance 1639
62.5 Driver Monitoring 1645
62.6 Towards Fully Autonomous Vehicles 1647
62.7 Future Trends and Prospects 1650
62.8 Conclusions and Further Reading 1651
Video-References 1651
References 1652

63 **Medical Robotics and Computer-Integrated Surgery**
Russell H. Taylor, Arianna Menciassi, Gabor Fichtinger, Paolo Fiorini, Paolo Dario 1657
63.1 Core Concepts 1658
63.2 Technology 1662
63.3 Systems, Research Areas, and Applications 1667
63.4 Conclusion and Future Directions 1675
Video-References 1676
References 1676

64 **Rehabilitation and Health Care Robotics**
H.F. Machiel Van der Loos, David J. Reinkensmeyer, Eugenio Guglielmelli . 1685
64.1 Overview 1686
64.2 Rehabilitation Therapy and Training Robots 1692
64.3 Aids for People with Disabilities 1703
64.4 Smart Prostheses and Orthoses 1711
64.5 Augmentation for Diagnosis and Monitoring 1713
64.6 Safety, Ethics, Access and Economics 1715
64.7 Conclusions and Further Readings 1716
Video-References 1717
References 1718

65 **Domestic Robotics**
Erwin Prassler, Mario E. Munich, Paolo Pirjanian, Kazuhiro Kosuge 1729
65.1 Mobile Domestic Robotics 1730
65.2 Enabling Technologies 1747
65.3 Smart Homes 1754
Video-References 1757
References 1757

66 **Robotics Competitions and Challenges**
Daniele Nardi, Jonathan Roberts, Manuela Veloso, Luke Fletcher 1759
66.1 Introduction 1760
66.2 Overview 1760
66.3 Competitions Inspired by Human Competitions 1762
66.4 Task-Oriented Competitions 1769
66.5 Conclusion and Further Reading 1780
Video-References 1781
References 1781

Part G Robots and Humans

67 **Humanoids**
Paul Fitzpatrick, Kensuke Harada, Charles C. Kemp, Yoshio Matsumoto, Kazuhito Yokoi, Eiichi Yoshida 1789
67.1 Why Humanoids? 1789
67.2 History 1792
67.3 What to Immitate? 1794
67.4 Locomotion 1795
67.5 Whole-Body Activities 1801
67.6 Morphological Communication 1809
67.7 Conclusions and Further Reading 1813
Video-References 1813
References 1813

68 **Human Motion Reconstruction**
Katsu Yamane, Wataru Takano 1819
68.1 Overview 1819
68.2 Models and Computations 1820
68.3 Reconstruction for Understanding 1825
68.4 Reconstruction for Robots 1829
Video-References 1830
References 1831

69 **Physical Human–Robot Interaction**
Sami Haddadin, Elizabeth Croft 1835
69.1 Classification 1836
69.2 Human Safety 1839
69.3 Human-Friendly Robot Design 1847
69.4 Control for Physical Interaction 1853
69.5 Motion Planning for Human Environments 1859
69.6 Interaction Planning 1862

69.7 Conclusions and Challenges 1867
Video-References 1868
References 1869

70 **Human–Robot Augmentation**
Massimo Bergamasco, Hugh Herr 1875
70.1 Concept and Definitions 1876
70.2 Upper Limb Wearable Systems 1877
70.3 Lower Limb Wearable Systems 1882
70.4 Whole Body Wearable Systems 1889
70.5 Control of Human–Robot Augmentation Systems 1892
70.6 Conclusions and Further Developments 1902
Video-References 1902
References 1902

71 **Cognitive Human–Robot Interaction**
Bilge Mutlu, Nicholas Roy, Selma Šabanović 1907
71.1 Human Models of Interaction 1908
71.2 Robot Models of Interaction 1914
71.3 Models of Human–Robot Interaction 1916
71.4 Conclusion and Further Reading 1927
Video-References 1927
References 1928

72 **Social Robotics**
Cynthia Breazeal, Kerstin Dautenhahn, Takayuki Kanda 1935
72.1 Overview 1936
72.2 Social Robot Embodiment 1936
72.3 Social Robots and Social-Emotional Intelligence 1938
72.4 Socio-Cognitive Skills 1941
72.5 Human Social Responses to Social Robots 1944
72.6 Social Robots and Communication Skills 1946
72.7 Long-Term Interaction with Robot Companions 1950
72.8 Tactile Interaction with Social Robots 1954
72.9 Social Robots and Teamwork 1958
72.10 Conclusion 1959
72.11 Further Reading 1960
Video-References 1960
References 1961

73 **Socially Assistive Robotics**
Maja J. Matarić, Brian Scassellati 1973
73.1 Overview 1973
73.2 The Need for Socially Assistive Robotics 1974
73.3 Advantages of Embodied Robots over Virtual Agents 1975
73.4 Motivation, Autonomy, and Companionship 1977
73.5 Influence and the Dynamics of Assistive Interaction 1978
73.6 Personalization and Adaptation to Specific Needs and Abilities 1978
73.7 Creating Long-Term Engagement and Behaviour Change 1979
73.8 SAR for Autism Spectrum Disorder (ASD) Therapy 1980
73.9 SAR Supporting Rehabilitation 1982

73.10 SAR and Eldercare 1985
73.11 SAR for Alzheimer's Dementia and Cognitive Rehabilitation 1986
73.12 Ethical and Safety Considerations 1987
References 1988

74 **Learning from Humans**
Aude G. Billard, Sylvain Calinon, Rüdiger Dillmann 1995
74.1 Learning of Robots 1995
74.2 Key Issues When Learning from Human Demonstrations 1998
74.3 Interfaces for Demonstration 2000
74.4 Algorithms to Learn from Humans 2002
74.5 Conclusions and Open Issues in Robot LfD 2008
Video-References 2009
References 2009

75 **Biologically Inspired Robotics**
Fumiya Iida, Auke Jan Ijspeert 2015
75.1 General Background 2016
75.2 Methodology 2017
75.3 Case Studies 2021
75.4 Landscape of Bio-Inspired Robotics Research and Challenges 2026
75.5 Conclusion 2028
Video-References 2028
References 2029

76 **Evolutionary Robotics**
Stefano Nolfi, Josh Bongard, Phil Husbands, Dario Floreano 2035
76.1 Method 2036
76.2 First Steps 2036
76.3 Simulation and Reality 2040
76.4 Behavior as a Complex Adaptive System 2041
76.5 Evolving Bodies 2044
76.6 Seeing the Light 2046
76.7 Computational Neuroethology 2049
76.8 Evolution and Learning 2054
76.9 Evolution of Social Behavior 2057
76.10 Evolutionary Hardware 2060
76.11 Closing Remarks 2061
Video-References 2061
References 2062

77 **Neurorobotics: From Vision to Action**
Patrick van der Smagt, Michael A. Arbib, Giorgio Metta 2069
77.1 Definitions and History 2070
77.2 The Case for Vision 2071
77.3 Vertebrate Motor Control 2075
77.4 The Role of Mirror Systems 2082
77.5 Conclusion and Further Reading 2089
References 2090

78 **Perceptual Robotics**
Heinrich Bülthoff, Christian Wallraven, Martin A. Giese 2095
78.1 Perceptual Mechanisms of Object Representations 2097
78.2 Perceptual Mechanisms of Action Representation 2103
78.3 Perceptual Validation of Robotics 2107
78.4 Conclusion and Further Reading 2108
Video-References 2109
References 2109

79 **Robotics for Education**
David P. Miller, Illah Nourbakhsh 2115
79.1 The Role of Robots in Education 2116
79.2 Educational Robot Tournaments 2117
79.3 Education Robot Platforms 2120
79.4 Education Robot Controllers and Programming Environments 2123
79.5 Robotic Technologies for Student Learning 2127
79.6 Educational Evaluation of Robot Programs 2129
79.7 Conclusions and Further Reading 2131
Video-References 2131
References 2131

80 **Roboethics: Social and Ethical Implications**
Gianmarco Veruggio, Fiorella Operto, George Bekey 2135
80.1 A Methodological Note 2137
80.2 Specificity of Robotics 2138
80.3 Cultural Differences in the Acceptance of Robots 2138
80.4 Roboethics Foreshadowed in the Literature 2139
80.5 And Expressed in Real Robotics 2139
80.6 Ethics in Science and Technology 2140
80.7 Ethical Issues in an ICT Society 2143
80.8 Human Principles and Rights 2144
80.9 Legal Issues in Robotics 2146
80.10 Roboethics Taxonomy 2147
80.11 Roboethics Enforced: From Ideals to Rules 2156
80.12 Conclusions and Further Reading 2157
Video-References 2158
References 2159

Acknowledgements 2161
About the Authors 2163
Index 2197

Abbreviations

Symbols

k-NN	*k*-nearest neighbor
0-D	zero-dimensional
1-D	one-dimensional
2-D	two-dimensional
2.5-D	two-and-a-half-dimensional
3-D	three-dimensional
3-D-NDT	three-dimensional!normal distributions transform
4-D	four-dimensional
6-D	six-dimensional
6R	six-revolute
7R	seven-revolute

A

A&F	agriculture and forestry
AA	agonist–antagonist
AAAI	American Association for Artificial Intelligence
AAAI	Association for the Advancement of Artificial Intelligence
AAL	ambient assisted living
ABA	articulated-body algorithm
ABF	artificial bacterial flagella
ABRT	automated!bus rapid transit
ABS	acrylonitrile–butadiene–styrene
AC	aerodynamic center
AC	alternating current
ACARP	Australian Coal Association Research Program
ACBS	automatic!constructions building system
ACC	adaptive cruise control
ACFV	autonomous!combat flying vehicle
ACM	active chord mechanism
ACM	active cord mechanism
ACT	anatomically correct testbed
ADAS	advanced driving assistance system
ADC	analog digital conveter
ADCP	acoustic Doppler current profiler
ADL	activities for daily living
ADSL	asymmetric digital subscriber line
AFC	alkaline fuel cell
AFC	armoured (or articulated) face conveyor
AFM	atomic force microscope
AFV	autonomous!flying vehicle
AGV	autonomous guided vehicle
AGV	automated!guided vehicle
AHRS	attitude and heading reference system
AHS	advanced highway system
AI	artificial intelligence
AIAA	American Institute of Aeronautics and Astronautics
AIM	assembly incidence matrix
AIP	air-independent power
AIP	anterior intraparietal sulcus
AIP	anterior interparietal area
AIS	artificial intelligence system
AIST	Institute of Advanced Industrial Science and Technology
AIST	Japan National Institute of Advanced Industrial Science and Technology
AIST	National Institute of Advanced Industrial Science and Technology (Japan)
AIT	anterior inferotemporal cortex
ALEX	active leg exoskeleton
AM	actuator for manipulation
AMASC	actuator with mechanically adjustable series compliance
AMC	Association for Computing Machinery
AMD	autonomous!mental development
AMM	audio-motor map
ANN	artificial neural network
AO	Arbeitsgemeinschaft für Ostheosynthesefragen
AOA	angle of attack
AP	antipersonnel
APF	annealed particle filter
APG	adjustable pattern generator
API	application programming interface
APOC	allowing dynamic selection and changes
AR	autoregressive
aRDnet	agile robot development network
ARM	Acorn RISC machine architecture
ARM	assistive!robot service manipulator
ARX	auto regressive estimator
ASAP	adaptive sampling and prediction
ASCII	American standard code for information interchange
ASD	autism spectrum disorder
ASIC	application-specific integrated circuit
ASIC	application-specific feature transform
ASIMO	advanced step in innovative mobility
ASK	amplitude shift keying
ASL	autonomous systems laboratory
ASM	advanced servomanipulator

ASN	active sensor network
ASR	automatic!spoken-language recognition
ASR	automatic!speech recognition
ASTRO	autonomous!space transport robotic operations
ASV	adaptive suspension vehicle
ASyMTRe	automated!synthesis of multirobot task solutions through software reconfiguration
AT	anti-tank mine
ATHLETE	all-terrain hex-legged extra-terrestrial explorer
ATLANTIS	a three layer architecture for navigating through intricate situations
ATLSS	advanced technology for large structural systems
ATR	automatic!target recognition
AuRA	autonomous robot architecture
AUV	autonomous underwater vehicle
AUV	autonomous aquatic vehicle
AUVAC	Autonomous Undersea Vehicles Application Center
AUVSI	Association for Unmanned Vehicle Systems International
AV	anti-vehicle

B

B/S	browser/server
B2B	business to business
BCI	brain-computer interface
BE	body extender
BEMT	blade element momentum theory
BEST	boosting!engineering science and technology
BET	blade element theory
BFA	bending fluidic actuator
BFP	best-first-planner
BI	brain imaging
BIP	behavior-interaction-priority
BLE	broadcast of local eligibility
BLEEX	Berkely exoskeleton
BLUE	best linear unbiased estimator
BML	behavior!mark-up language
BMS	battery management system
BN	Bayesian network
BOM	bill of material
BOw	bag-of-word
BP	behavior primitive
BP	base plate
BRICS	best practice in robotics
BRT	bus rapid transit
BWSTT	body-weight supported treadmill training

C

C	cylindrical joint
C/A	coarse-acquisition
C/S	client/server
CA	collision avoidance
CACC	cooperative adaptive cruise control
CAD	computer-aided drafting
CAD	computer-aided design
CAE	computer-aided engineering
CALM	communication access for land mobiles
CAM	computer-aided manufacturing
CAN	controller area network
CARD	computer-aided remote driving
CARE	coordination action for robotics in Europe
CASA	Civil Aviation Safety Authority
CASALA	Centre for Affective Solutions for Ambient Living Awareness
CASPER	continuous activity scheduling, planning, execution and replanning
CAT	collision avoidance technology
CAT	computer-aided tomography
CB	computional brain
CB	cluster bomb
CBRNE	chemical, biological, nuclear, radiological, or explosive
CC	compression criterion
CCD	charge-coupled device
CCD	charge-coupled detector
CCI	control command interpreter
CCP	coverage configuration protocol
CCT	conservative congruence transformation
CCW	counterclockwise
CC&D	camouflage, concealment, and deception
CD	collision detection
CD	committee draft
CD	compact disc
CDC	cardinal direction calculus
CDOM	colored dissolved organic matter
CE	computer ethic
CEA	Commissariat à l'Énergie Atomique
CEA	Atomic Energy Commission
CEBOT	cellular robotic system
CEC	Congress on Evolutionary Computation
CEPE	Computer Ethics Philosophical Enquiry
CES	Consumer Electronics Show
CF	carbon fiber
CF	contact formation
CF	climbing fiber
CFD	computational fluid dynamics
CFRP	carbon fiber reinforced prepreg
CFRP	carbon fiber reinforced plastic
CG	center of gravity

CG	computer graphics
CGI	common gateway interface
CHMM	coupled!hidden Markov model
CHMM	continuous hidden Markov model
CIC	computer integrated construction
CIE	International Commission on Illumination
CIP	Children's Innovation Project
CIRCA	cooperative intelligent real-time control architecture
CIS	computer-integrated surgery
CLARAty	coupled layered architecture for robot autonomy
CLEaR	closed-loop execution and recovery
CLIK	closed-loop inverse kinematics
CMAC	cerebellar model articulation controller
CMCs	ceramic matrix composite
CML	concurrent!mapping and localization
CMM	coordinate measurement machine
CMOMMT	cooperative multirobot observation of multiple moving target
CMOS	complementary metal-oxide-semiconductor
CMP	centroid moment pivot
CMTE	Cooperative Research Centre for Mining Technology and Equipment
CMU	Carnegie Mellon University
CNC	computer numerical control
CNN	convolutional neural network
CNP	contract net protocol
CNRS	Centre National de la Recherche Scientifique
CNT	carbon nanotube
COCO	common objects in context
COG	center of gravity
COM	center of mass
COMAN	compliant humanoid platform
COMEST	Commission mondiale d'éthique des connaissances scientifiques et des technologies
COMINT	communication intelligence
CONE	Collaborative Observatory for Nature Environments
COP	center of pressure
CoP	center of pressure
COR	center of rotation
CORBA	common object request broker architecture
CORS	continuous operating reference station
COT	cost!of transport
COTS	commercial off-the-shelf
COV	characteristic output vector
CP	complementarity problem
CP	capture point
CP	continuous path
CP	cerebral palsy
CPG	central pattern generation
CPG	central pattern generator
CPS	cyber physical system
CPSR	Computer Professional for Social Responsibility
CPU	central processing unit
CRASAR	Center for Robot-Assisted Search and Rescue
CRBA	composite-rigid-body algorithm
CRF	conditional random field
CRLB	Cramér–Rao lower bound
CSAIL	Computer Science and Artificial Intelligence Laboratory
CSIRO	Commonwealth Scientific and Industrial Research Organisation
CSMA	carrier-sense multiple-access
CSP	constraint satisfaction problem
CSSF	Canadian Scientific Submersile Facility
CT	computed tomography
CTFM	continuous-transmission frequency modulation
CU	control unit
cv-SLAM	ceiling vision SLAM
CVD	chemical vapor deposition
CVIS	cooperative vehicle infrastructure system
CVT	continuous variable transmission
CW	clockwise
CWS	contact!wrench sum

D

D	distal
D/A	digital-to-analog
DAC	digital analog converter
DARPA	Defense Advanced Research Projects Agency
DARS	distributed!autonomous robotic systems
DBN	dynamic Bayesian network
DBN	deep belief network
DC	disconnected
DC	direct current
DC	dynamic!constrained
DCS	dynamic covariance scaling
DCT	discrete!cosine transform
DD	differentially driven
DDD	dangerous, dirty, and dreary
DDF	decentralized data fusion
DDP	differential dynamic programming
DDS	data distribution service
DEA	differential elastic actuator
DEM	discrete!element method

DFA	design!for assembly
DFRA	distributed field robot architecture
DFT	discrete Fourier transform
DGPS	differential global positioning system
DH	Denavit–Hartenberg
DHMM	discrete!hidden Markov model
DHS	US Department of Homeland Security
DIRA	distributed!robot architecture
DIST	Dipartmento di Informatica Sistemica e Telematica
DL	description logic
DLR	Deutsches Zentrum für Luft- und Raumfahrt
DLR	German Aerospace Center
DMFC	direct methanol fuel cell
DMP	dynamic movement primitive
DNA	deoxyribonucleic acid
DNF	dynamic!neural field
DOD	Department of Defense
DOF	degree of freedom
DOG	difference of Gaussian
DOP	dilution of precision
DPLL	Davis–Putnam algorithm
DPM	deformable part model
DPN	dip-pen nanolithography
DPSK	differential phase shift keying
DRIE	deep reactive ion etching
DSM	dynamic!state machine
DSO	Defense Sciences Office
DSP	digital signal processor
DSRC	dedicated short-range communications
DU	dynamic!unconstrained
DVL	Doppler velocity log
DWA	dynamic window approach
DWDM	dense wave division multiplex
D&D	deactivation and decommissioning

E

e-beam	electron-beam
EAP	electroactive polymer
EBA	energy bounding algorithm
EBA	extrastriate body part area
EBID	electron-beam induced deposition
EC	externally connected
EC	exteroception
ECAI	European Conference on Artificial Intelligence
ECD	eddy current damper
ECEF	earth-centred, earth-fixed
ECER	European Conference on Educational Robotics
ECG	electrocardiogram
ECU	electronics controller unit
EDM	electrical discharge machining
EE	end-effector
EEG	electroencephalography
EGNOS	European Geostationary Navigation Overlay Service
EHC	enhanced horizon control
EHPA	exoskeleton!for human performance augmentation
EKF	extended Kalman filter
ELS	ethical, legal and societal
EM	expectation maximization
emf	electromotive force
EMG	electromyography
EMIB	emotion, motivation and intentional behavior
EMS	electrical!master–slave manipulator
EO	electrooptical
EO	elementary operator
EOA	end of arm
EOD	explosive!ordnance disposal
EP	exploratory procedure
EP	energy packet
EPFL	Ecole Polytechnique Fédérale de Lausanne
EPP	extended!physiological proprioception
EPS	expandable polystyrene
ER	electrorheological
ER	evolutionary!robotics
ERA	European robotic arm
ERP	enterprise resource planning
ERSP	evolution robotics software platform
ES	electrical!stimulation
ESA	European Space Agency
ESC	electronic speed controller
ESL	execution support language
ESM	energy!stability margin
ESM	electric support measure
ETL	Electro-Technical Laboratory
ETS-VII	Engineering Test Satellite VII
EU	European Union
EURON	European Robotics Research Network
EVA	extravehicular activity
EVRYON	evolving morphologies for human–robot symbiotic interaction

F

F5	frontal area 5
FAA	Federal Aviation Administration
FAO	Food and Agriculture Organization
FARS	Fagg–Arbib–Rizzolatti–Sakata
FARSA	framework for autonomous robotics simulation and analysis

fastSLAM	fast simultaneous localization and mapping
FB-EHPA	full-body EHPA
FCU	flight control-unit
FD	friction damper
FDA	US Food and Drug Association
FDM	fused deposition modeling
FE	finite element
FEA	finite element analysis
FEM	finite element method
FESEM	field-emission SEM
FF	fast forward
FFI	Norwegian defense research establishment
FFT	fast Fourier transform
FIFO	first-in first-out
FIRA	Federation of International Robot-soccer Association
FIRRE	family of integrated rapid response equipment
FIRST	For Inspiration and Recognition of Science and Technology
Fl-UAS	flapping wing unmanned aerial system
FLIR	forward!looking infrared
FMBT	feasible minimum buffering time
FMCW	frequency modulation continuous wave
fMRI	functional!magnetic resonance imaging
FMS	flexible!manufacturing system
FNS	functional!neural stimulation
FOA	focus of attention
FOG	fiber-optic gyro
FOPEN	foliage penetration
FOPL	first-order predicate logic
FOV	field of view
FP	fusion primitive
FPGA	field-programmable gate array
FR	false range
FRI	foot rotation indicator
FRP	fiber-reinforced plastics
FRP	fiber-reinforced prepreg
fs	force!sensor
FSA	finite-state acceptor
FSK	frequency shift keying
FSR	force sensing resistor
FSW	friction!stir welding
FTTH	fiber to the home
FW	fixed-wing

G

GA	genetic algorithm
GAPP	goal as parallel programs
GARNICS	gardening with a cognitive system
GAS	global asymptotic stability
GBAS	ground based augmentation system
GCDC	Grand Cooperative Driving Challenge
GCER	Global Conference on Educational Robotics
GCR	goal-contact relaxation
GCS	ground!control station
GDP	gross!domestic product
GenoM	generator of modules
GEO	geostationary Earth orbit
GF	grapple fixture
GFRP	glass-fiber reinforced plastic
GI	gastrointestinal
GIB	GPS intelligent buoys
GICHD	Geneva International Centre for Humanitarian Demining
GID	geometric!intersection data
GIE	generalized-inertia ellipsoid
GIS	geographic information system
GJM	generalized!Jacobian matrix
GLONASS	globalnaya navigatsionnaya sputnikovaya sistema
GLS	global navigation satellite system
GMAW	gas-shielded metal arc welding
GMM	Gaussian mixture model
GMSK	Gaussian minimum shift keying
GMTI	ground!moving target indicator
GNC	guidance, navigation, and control
GO	golgi!tendon organ
GP	Gaussian process
GPCA	generalized principal component analysis
GPRS	general!packet radio service
GPS	global positioning system
GPU	graphics processing unit
GRAB	guaranteed recursive adaptive bounding
GRACE	graduate robot attending conference
GraWoLF	gradient-based win or learn fast
GSD	geon structural description
GSN	gait sensitivity norm
GSP	Gough–Stewart platform
GUI	graphical user interface
GV	ground vehicle
GVA	gross!value added
GZMP	generalized!ZMP

H

H	helical joint
HAL	hybrid!assistive limb
HAMMER	hierarchical!attentive multiple models for execution and recognition
HASY	hand!arm system
HBBA	hybrid behavior-based architecture
HCI	human–computer interaction

HD	high definition
HD	haptic device
HD-SDI	high-definition serial digital interface
HDSL	high data rate digital subscriber line
HE	hand!exoskeleton
HF	hard finger
HF	histogram filter
HFAC	high frequency alternating current
HHMM	hierarchical!hidden Markov model
HIC	head injury criterion
HIII	Hybrid III dummy
HIP	haptic interaction point
HJB	Hamilton–Jacobi–Bellman
HJI	Hamilton–Jacobi–Isaac
HMCS	human–machine!cooperative system
HMD	head-mounted display
HMDS	hexamethyldisilazane
HMI	human–machine!interaction
HMI	human–machine!interface
HMM	hidden Markov model
HO	human operator
HOG	histogram of oriented gradient
HOG	histogram of oriented features
HPC	high-performance computing
HRI	human–robot interaction
HRI/OS	HRI operating system
HRP	humanoid robotics project
HRR	high resolution radar
HRTEM	high-resolution transmission electron microscope
HSGR	high safety goal
HST	Hubble space telescope
HSTAMIDS	handheld standoff mine detection system
HSWR	high safety wide region
HTAS	high tech automotive system
HTML	hypertext markup language
HTN	hierarchical task network
HTTP	hypertext transmission protocol
HW/SW	hardware/software

I

I/O	input/output
I3CON	industrialized, integrated, intelligent, construction
IA	interval algebra
IA	instantaneous!allocation
IAA	interaction!agent
IAB	International Association of Bioethics
IACAP	International Association for Computing and Philosophy
IAD	interaural amplitude difference
IAD	intelligent!assisting device
IARC	International Aerial Robotics Competition
IAS	intelligent!autonomous system
IBVS	image-based visual servo control
IC	integrated chip
IC	integrated circuit
ICA	independent!component analysis
ICAPS	International Conference on Automated Planning and Scheduling
ICAR	International Conference on Advanced Robotics
ICBL	International Campaign to Ban Landmines
ICC	instantaneous center of curvature
ICE	internet communications engine
ICP	iterative closest point
ICR	instantaneous center of rotation
ICRA	International Conference on Robotics and Automation
ICT	information!and communication technology
ID	inside diameter
ID	identifier
IDE	integrated!development environment
IDL	interface definition language
IE	information!ethics
IED	improvised explosive device
IEEE	Institute of Electrical and Electronics Engineers
IEKF	iterated extended Kalman filter
IETF	internet!engineering task force
IFA	Internationale Funk Ausstellung
IFOG	interferometric fiber-optic gyro
IFR	International Federation of Robotics
IFREMER	Institut français de recherche pour l'exploitation de la mer
IFRR	International Foundation of Robotics Research
IFSAR	interferometric SAR
IHIP	intermediate haptic interaction point
IIR	infinite impulse response
IIS	Internet Information Services
IIT	Istituto Italiano di Tecnologia
IJCAI	International Joint Conference on Artificial Intelligence
IK	inverse kinematics
ILLS	instrumented logical sensor system
ILO	International Labor Organization
ILQR	iterative linear quadratic regulator
IM	injury measure
IMAV	International Micro Air Vehicles
IMTS	intelligent!multimode transit system
IMU	inertial measurement unit
INS	inertia navigation system

INS	inertial navigation system
IO	input output
IO	inferior olive
IOSS	input-output-to-state stability
IP	internet protocol
IP	interphalangeal
IPA	Institute for Manufacturing Engineering and Automation
IPC	interprocess communication
IPC	international AI planning competition
IPMC	ionic polymer-metal composite
IPR	intellectual property right
IR	infrared
IRB	Institutional Review Board
IREDES	International Rock Excavation Data Exchange Standard
IRL	in real life
IRL	inverse!reinforcement learning
IRLS	iteratively reweighted least square
IRNSS	Indian regional navigational satellite system
IROS	Intelligent Robots and Systems
IS	importance sampling
ISA	industrial standard architecture
ISA	international standard atmosphere
ISAR	inverse SAR
ISDN	integrated services digital network
ISE	international submarine engineering
ISER	International Symposium on Experimental Robotics
ISM	implicit shape model
ISO	International Organization for Standardization
ISP	Internet service provider
ISR	intelligence, surveillance and reconnaissance
ISRR	International Symposium of Robotics Research
ISS	international space station
ISS	input-to-state stability
IST	Instituto Superior Técnico
IST	Information Society Technologies
IT	intrinsic tactile
IT	information!technology
IT	inferotemporal cortex
ITD	interaural time difference
IU	interaction!unit
IV	instrumental variable
IvP	interval programming
IWS	intelligent!wheelchair system
IxTeT	indexed time table

J

JAEA	Japan Atomic Energy Agency
JAMSTEC	Japan Agency for Marine-Earth Science and Technology
JAMSTEC	Japan Marine Science and Technology Center
JAUS	joint architecture for unmanned systems
JAXA	Japan Aerospace Exploration Agency
JDL	joint directors of laboratories
JEM	Japan Experiment Module
JEMRMS	Japanese experiment module remote manipulator system
JHU	Johns Hopkins University
JND	just noticeable difference
JPL	Jet Propulsion Laboratory
JPS	jigsaw positioning system
JSC	Johnson Space Center
JSIM	joint-space inertia matrix
JSP	Java server pages

K

KAIST	Korea Advanced Institute of Science and Technology
KERS	kinetic energy recovery system
KIPR	KISS Institute for Practical Robotics
KLD	Kullback–Leibler divergence
KNN	k-nearest neighbor
KR	knowledge representation
KRISO	Korea Research Institute of Ships and Ocean Engineering

L

L/D	lift-to-drag
LAAS	Laboratory for Analysis and Architecture of Systems
LADAR	laser radar
LAGR	learning!applied to ground robots
LARC	Lie algebra rank condition
LARS	Laparoscopic Assistant Robotic System
LASC	Longwall Automation Steering Committee
LBL	long-baseline system
LCAUV	long-range cruising AUV
LCC	life-cycle-costing
LCD	liquid-crystal display
LCM	light-weight communications and marshalling
LCP	linear complementarity problem
LCSP	linear constraint satisfaction program
LDA	latent Dirichlet allocation
LED	light-emitting diode
LENAR	lower!extremity nonanthropomorphic robot

LEO	low!Earth orbit
LEV	leading edge vortex
LfD	learning!from demonstration
LGN	lateral!geniculate nucleus
LHD	load!haul-dump
LIDAR	light detection and ranging
LIGA	Lithographie, Galvanoumformung, Abformung
LIP	linear inverted pendulum
LIP	lateral!intraparietal sulcus
LiPo	lithium polymer
LLC	locality constrained linear coding
LMedS	least median of squares
LMS	laser measurement system
LOG	Laplacian of Gaussian
LOPES	lower!extremity powered exoskeleton
LOS	line-of-sight
LP	linear program
LQG	linear quadratic Gaussian
LQR	linear quadratic regulator
LSS	logical sensor system
LSVM	latent support vector machine
LtA	lighter-than-air
LtA-UAS	lighter-than-air system
LTL	linear temporal logic
LVDT	linear variable differential transformer
LWR	light-weight robot

M

MACA	Afghanistan Mine Action Center
MACCEPA	mechanically adjustable compliance and controllable equilibrium position actuator
MAP	maximum a posteriori
MARS	multiappendage robotic system
MARUM	Zentrum für Marine Umweltwissenschaften
MASE	Marine Autonomous Systems Engineering
MASINT	measurement!and signatures intelligence
MAV	micro aerial vehicles
MAZE	Micro robot maze contest
MBA	motivated behavioral architecture
MBARI	Monterey Bay Aquarium Research Institute
MBE	molecular-beam epitaxy
MBS	mobile!base system
MC	Monte Carlo
MCFC	molten carbonate fuel cell
MCP	magazining, cleaning, plotting
MCP	metacarpophalangeal
MCS	mission!control system
MDARS	mobile!detection assessment and response system
MDL	minimum description length
MDP	Markov decision process
ME	mechanical!engeneering
MEG	magnetoencephalography
MEL	Mechanical Engineering Laboratory
MEMS	microelectromechanical system
MEP	motor!evoked potential
MESSIE	multi expert system for scene interpretation and evaluation
MESUR	Mars environmental survey
MF	mossy fiber
MFI	micromechanical flying insect
MFSK	multiple FSK
MHS	International Symposium on Micro Mechatronics and Human Science
MHT	multihypothesis tracking
MIA	mechanical impedance adjuster
MIME	mirror!image movement enhancer
MIMICS	multimodal immersive motion rehabilitation with interactive cognitive system
MIMO	multiple-input–multiple-output
MIP	medial intraparietal sulcus
MIPS	microprocessor without interlocked pipeline stages
MIR	mode identification and recovery
MIRO	middleware for robot
MIS	minimally invasive surgery
MIT	Massachusetts Institute of Technology
MITI	Ministry of International Trade and Industry
MKL	multiple kernel learning
ML	machine!learning
MLE	maximum likelihood estimate
MLR	mesencephalic locomotor region
MLS	multilevel surface map
MMC	metal matrix composite
MMMS	multiple master multiple-slave
MMSAE	multiple model switching adaptive estimator
MMSE	minimum mean-square error
MMSS	multiple master single-slave
MNS	mirror!neuron system
MOCVD	metallo-organic chemical vapor deposition
MOMR	multiple operator multiple robot
MOOS	mission oriented operating suite
MOOS	motion-oriented operating system
MORO	mobile robot
MOSR	multiple operator single robot
MP	moving plate
MPC	model predictive control

MPF	manifold particle filter
MPFIM	multiple!paired forward-inverse model
MPHE	multiphalanx hand exoskeleton
MPSK	Mary phase shift keying
MQAM	Mary quadrature amplitude modulation
MR	magnetorheological
MR	multiple reflection
MR	multirobot!task
MRAC	model reference adaptive control
MRDS	Microsoft robotics developers studio
MRF	Markov random field
MRHA	multiple!resource host architecture
MRI	magnetic resonance imaging
MRSR	Mars rover sample return
MRTA	multirobot!task allocation
MSAS	multifunctional satellite augmentation system
MSER	maximally stable extremal region
MSHA	US Mine Safety and Health Administration
MSK	minimum shift keying
MSL	middle-size league
MSM	master–slave!manipulator
MST	microsystem technology
MT	momentum theory
MT	multitask
MT	medial temporal area
MTBF	mean time between failures
MTI	moving target indicator
MVERT	move value estimation for robot teams
MWNT	multiwalled carbon nanotube

N

N&G	nursery and greenhouse
NAP	nonaccidental property
NASA	National Aeronautics and Space Agency
NASDA	National Space Development Agency of Japan
NASREM	NASA/NBS standard reference model
NBS	National Bureau of Standards
NC	numerical control
ND	nearness diagram navigation
NDDS	network data distribution service
NDGPS	nationwide different GPS system
NDI	nonlinear dynamic inversion
NDT	normal distributions transform
NEMO	network!mobility
NEMS	nanoelectromechanical system
NEO	neodymium
NERVE	New England Robotics Validation and Experimentation
NESM	normalized ESM
NIDRR	National Institute on Disability and Rehabilitation Research
NiMH	nickel metal hydride battery
NIMS	networked!infomechanical systems
NIOSH	United States National Institute for Occupational Safety and Health
NIRS	near infrared spectroscopy
NIST	National Institute of Standards and Technology
NLIS	national livestock identification scheme
NLP	nonlinear!programming problem
NMEA	National Marine Electronics Association
NMF	nonnegative matrix factorization
NMMI	natural machine motion initiative
NMR	nuclear!magnetic resonance
NN	neural network
NOAA	National Oceanic and Atmospheric Administration
NOAH	navigation!and obstacle avoidance help
NOC	National Oceanography Centre
NOTES	natural!orifice transluminal surgery
NPO	nonprofit organization
NPS	Naval Postgraduate School
NQE	national qualifying event
NRI	national robotics initiative
NRM	nanorobotic manipulator
NRTK	network real-time kinematic
NTPP	nontangential proper part
NTSC	National Television System Committee
NURBS	nonuniform rational B-spline
NUWC	Naval Undersea Warfare Center Division Newport
NZDF	New Zealand Defence Force

O

OAA	open!agent architecture
OASIS	onboard autonomous science investigation system
OAT	optimal arbitrary time-delay
OBU	on board unit
OC	optimal control
OCPP	optimal!coverage path planning
OCR	OC robotics
OCT	optical!coherence tomography
OCU	operator control unit
OD	outer diameter
ODE	ordinary differential equation
ODE	open dynamics engine
ODI	ordinary differential inclusion
OECD	Organization for Economic Cooperation and Development
OKR	optokinetic response
OLP	offline programming

OM	optical microscope
OM	occupancy map
ONR	US Office of Naval Research
OOF	out of field
OOTL	human!out of the loop control
OPRoS	open platform for robotic service
ORCA	open robot control architecture
ORCCAD	open robot controller computer aided design
ORI	open!roboethics initiative
ORM	obstacle restriction method
OROCOS	open robot control software
ORU	orbital replacement unit
OS	operating system
OSC	operational-space control
OSIM	operational-space inertia matrix
OSU	Ohio State University
OTH	over-the-horizon
OUR-K	ontology based unified robot knowledge
OWL	web ontology language
OxIM	Oxford intelligent machine

P

P	prismatic joint
P&O	prosthetics!and orthotic
PA	point algebra
PACT	perception!for action control theory
PAD	pleasure arousal dominance
PAFC	phosphoric acid fuel cell
PAM	pneumatic artificial muscle
PaMini	pattern-based mixed-initiative
PANi	polyaniline
PANTOMEC	pantograph mechanism driven
PAPA	privacy, accuracy, intellectual property, and access
PAS	pseudo-amplitude scan
PAT	proximity!awareness technology
PB	parametric!bias
PbD	programming!by demonstration
PBVS	pose-based visual servo control
PC	polycarbonate
PC	personal computer
PC	principal contact
PC	passivity controller
PC	proprioception
PC	Purkinje cell
PCA	principal component analysis
PCI	peripheral component interconnect
PCIe	peripheral component interconnect express
PCL	point cloud library
PCM	programmable!construction machine
PD	proportional–derivative
PDE	partial differential equation
PDGF	power!data grapple fixture
PDMS	polydimethylsiloxane
PDOP	positional dilution of precision
PDT	proximity!detection technology
PEAS	probing environment and adaptive sleeping protocol
PEFC	polymer electrolyte fuel cell
PEMFC	proton exchange membrane fuel cell
PerceptOR	perception!for off-road robotics
PET	positron emission tomography
PF	particle filter
PF	parallel!fiber
PFC	prefrontal cortex
PFH	point feature histogram
PFM	potential field method
PGM	probabilistic graphical model
PGRL	policy gradient!reinforcement learning
pHRI	physical!human–robot interaction
PI	policy iteration
PI	possible!injury
PI	propositional integral
PI	proportional–integral
PIC	programmable!intelligent computer
PID	proportional–integral–derivative
PIT	posterior!inferotemporal cortex
PKM	parallel kinematics machine
PKM	parallel kinematic machine
PL	power loading
PLC	programmable!logic controller
PLD	programmable!logic device
PLEXIL	plan execution interchange language
PLSA	probabilistic latent semantic analysis
PLZT	lead lanthanum zirconate titanate
PM	permanent magnet
PMC	polymer matrix composite
PMMA	polymethyl methacrylate
PneuNet	pneumatic network
PnP	prespective-n-point
PNT	Petri net transducer
PO	partially overlapping
PO	passivity observer
POE	local product-of-exponential
POI	point!of interest
POM	polyoxymethylene
POMDP	partially observable Markov decision process
POP	partial-order planning
PPS	precise positioning system
PPy	polypyrrole
PR	positive photoresist
PRM	probabilistic roadmap
PRM	probabilistic roadmap method
PRN	pseudo-random noise

PRoP	personal roving presence
ProVAR	professional vocational assistive robot
PRS	procedural reasoning system
PS	power source
PSD	position sensing device
PSD	position-sensitive-device
PSK	phase shift keying
PSPM	passive set-position modulation
PTAM	parallel tracking and mapping
PTU	pan–tilt unit
PUMA	programmable!universal machine for assembly
PVA	position, velocity, and attitude
PVC	polyvinyl chloride
PVD	physical vapor deposition
PVDF	polyvinylidene fluoride
PWM	pulse-width modulation
PwoF	point-contact-without-friction
PZT	lead zirconate titanate

Q

QAM	quadrature amplitude modulation
QD	quantum dot
QID	qualifier, inspection and demonstration
QOLT	quality!of life technology
QOS	quality of service
QP	quadratic programming
QPSK	quadrature phase shift keying
QRIO	quest for curiosity
QSC	quasistatic!constrained
QT	quasistatic telerobotics
QZSS	quasi-zenith satellite system

R

R	revolute joint
R.U.R.	Rossum's Universal Robots
RA	rectangle algebra
RAC	Robotics and Automation Council
RAIM	receiver autonomous integrity monitor
RALF	robotic arm large and flexible
RALPH	rapidly adapting lane position handler
RAM	random!access memory
RAMS	robot-assisted microsurgery
RAMS	random!access memory system
RANSAC	random sample consensus
RAP	reactive action package
RAS	Robotics and Automation Society
RBC	recognition!by-component
RBF	radial!basis function network
RBF	radial!basis function
RBT	robot!experiment
RC	radio control
RC	robot!controller
RCC	region connection calculus
RCC	remote center of compliance
RCM	remote!center of motion
RCP	rover chassis prototype
RCR	responsible conduct of research
RCS	real-time control system
RCS	rig control system
RDT	rapidly exploring dense tree
RECS	robotic!explosive charging system
REINFORCE	reward increment = nonnegative factor × offset reinforcement × characteristic eligibility
RERC	Rehabilitation Engineering Research Center
RF	radio frequency
RFID	radio frequency identification
RG	rate gyro
RGB-D	color camera with depth
RGB-D	red green blue distance
RGB-D	red–green–blue–depth
RHIB	rigid!hull inflatable boat
RIE	reactive-ion etching
RIG	rate-integrating gyro
RISC	reduced instruction set computer
RL	reinforcement learning
RLG	ring laser gyroscope
RLG	random loop generator
RMC	resolved momentum control
RMDP	relational Markov decision processes
RMMS	reconfigurable modular manipulator system
RMS	root mean square
RNDF	route network definition file
RNEA	recursive Newton–Euler algorithm
RNN	recurrent neural network
RNNPB	recurrent neural network with parametric bias
RNS	reaction!null-space
ROC	receiver operating curve
ROC	remote!operations centre
ROCCO	robot!construction system for computer integrated construction
ROD	robot!oriented design
ROKVISS	robotics component verification on ISS
ROKVISS	robotics!components verification on the ISS
ROM	run-of-mine
ROM	read-only memory
ROMAN	Robot and Human Interactive Communication
ROS	robot operating system
ROV	remotely operated vehicle
ROV	remotely!operated underwater vehicle

RP		rapid prototyping
RP-VITA		remote presence virtual + independent telemedicine assistant
RPC		remote procedure call
RPI		Rensselaer Polytechnic Institute
RPS		room positioning system
RRSD		Robotics and Remote Systems Division
RRT		rapidly exploring random tree
RS		Reeds and Shepp
RSJ		Robotics Society of Japan
RSS		Robotics Science and Systems
RSTA		reconnaissance, surveillance, and target acquisition
RSU		road!side unit
RT		real-time
RT		room temperature
RT		reaction!time
RTCMS	C104	Radio Technical Commission for Maritime Services Special Committee 104
RTD		resistance temperature devices
RTI		real-time innovation
RTK		real-time kinematics
rTMS		repetitive!TMS
RTS		real-time system
RTT		real-time toolkit
RV		rotary vector
RVD		rendezvous/docking
RW		rotary-wing
RWI		real-world interface
RWS		robotic workstation
R&D		research and development
R&D		research and development

S

SA	simulated annealing
SA	selective availability
SAFMC	Singapore Amazing Flying Machine Competition
SAI	simulation!and active interfaces
SAM	smoothing and mapping
SAN	semiautonomous navigation
SAR	synthetic aperture radar
SAR	socially assistive robotics
SARSA	state action-reward-state-action
SAS	synthetic aperture sonar
SAS	stability augmentation system
SAT	International Conference on Theory and Applications of Satisfiability Testing
SBAS	satellite-based augmentation system
SBL	short baseline
SBSS	space based space surveillance
SC	sparse coding
SCARA	selective compliance assembly robot arm
SCI	spinal cord!injury
sci-fi	science fiction
SCM	smart composite microstructures
SCM	soil!contact model
SD	standard deviation
SDK	standard development kit
SDK	software development kit
SDM	shape deposition manufacturing
SDR	software!for distributed robotics
SDV	spatial dynamic voting
SEA	series elastic actuator
SEE	standard!end effector
SELF	sensorized environment for life
SEM	scanning electron microscope
SET	single electron transistor
SF	soft finger
SFM	structure from motion
SFX	sensor fusion effect
SGAS	semiglobal asymptotic stability
SGD	stochastic gradient descent
SGM	semiglobal!matching
SGUUB	semiglobal uniform ultimate boundedness
SIFT	scale-invariant feature transform
SIGINT	signal!intelligence
SIR	sampling importance resampling
SISO	single input single-output
SKM	serial!kinematic machines
SLA	stereolithography
SLAM	simultaneous localization and mapping
SLICE	specification language for ICE
SLIP	spring loaded inverted pendulum
SLRV	surveyor lunar rover vehicle
SLS	selective laser sintering
SM	static margin
SMA	shape memory alloy
SMAS	solid material assembly system
SMC	sequential Monte Carlo
SME	small!and medium enterprises
SMMS	single-master multiple-slave
SMP	shape memory polymer
SMS	short message service
SMSS	single-master single-slave
SMT	satisfiabiliy modulo theory
SMU	safe!motion unit
SNAME	society of naval architects and marine engineer
SNOM	scanning near-field optical microscopy
SNR	signal-to-noise ratio
SNS	spallation neutron source
SOFC	solid oxide fuel cell
SOI	silicon-on-insulator

SOMA	stream-oriented messaging architecture
SOMR	single operator multiple robot
SOS	save our souls
SOSR	single operator single robot
SPA	sense-plan-act
SPaT	signal!phase and timing
SPAWAR	Space and Naval Warfare Systems Center
SPC	self-posture changeability
SPDM	special purpose dexterous manipulator
SPHE	single-phalanx hand exoskeleton
SPL	single!port laparoscopy
SPL	standard!platform
SPM	scanning probe microscope
SPM	spatial pyramid matching
SPMS	shearer position measurement system
SPS	standard position system
SPU	spherical, prismatic, universal
SQP	sequential!quadratic programming
SR	single-robot task
SRA	spatial!reasoning agent
SRCC	spatial remote center compliance
SRI	Stanford Research Institute
SRMS	shuttle remote manipulator system
SSA	sparse surface adjustment
SSC	smart soft composite
SSL	small-size league
SSRMS	space!station remote manipulator system
ST	single-task
STEM	science, technology, engineering and mathematics
STM	scanning tunneling microscope
STP	simple temporal problem
STriDER	self-excited tripodal dynamic experimental robot
STS	superior!temporal sulcus
SUGV	small!unmanned ground vehicle
SUN	scene understanding
SURF	robust feature
SVD	singular value decomposition
SVM	support vector machine
SVR	support vector regression
SWNT	single-walled carbon nanotube
SWRI	Southwest Research Institute

T

T-REX	teleo-reactive executive
TA	time-extended assignment
TAL	temporal action logic
TAM	taxon!affordance model
TAP	test action pair
TBG	time-base generator
TC	technical committee
TCFFHRC	Trinity College's Firefighting Robot Contest
TCP	transfer control protocol
TCP	tool center point
TCP	transmission control protocol
TCSP	temporal constraint satisfaction problem
tDCS	transcranial!direct current stimulation
TDL	task description language
TDT	tension-differential type
TECS	total energy control system
TEM	transmission electron microscope
tEODor	telerob explosive ordnance disposal and observation robot
TFP	total!factor productivity
TL	temporal logic
TMM	transfer matrix method
TMS	tether management system
TMS	transcranial!magnetic stimulation
TNT	trinitrotoluene
TOA	time of arrival
TOF	time-of-flight
ToF	time-of-flight
TORO	torque!controlled humanoid robot
TPaD	tactile pattern display
TPBVP	two-point boundary value problem
TPP	tangential proper part
TRC	Transportation Research Center
TRIC	task space retrieval using inverse optimal control
TS	technical!specification
TSEE	teleoperated!small emplacement excavator
TSP	telesensor programming
TTC	time-to-collision
TUM	Technical University of Munich
TV	television

U

U	universal joint
UAS	unmanned aircraft system
UAS	unmanned!aerial system
UAV	unmanned aerial vehicle
UAV	fusing air vehicle
UAV	fielded unmanned aerial vehicle
UB	University of Bologna
UBC	University of British Columbia
UBM	Universität der Bundeswehr Munich
UCLA	University of California, Los Angeles
UCO	uniformly completely observable
UDP	user datagram protocol
UDP	user data protocol
UGV	unmanned!ground vehicle
UHD	ultrahigh definition

UHF	ultrahigh frequency
UHV	ultrahigh-vacuum
UKF	unscented Kalman filter
ULE	upper!limb exoskeleton
UML	unified modeling language
UMV	unmanned marine vehicle
UNESCO	United Nations Educational, Scientific and Cultural Organization
UPnP	universal plug and play
URC	Ubiquitous Robotic Companion
URL	uniform resource locator
USAR	urban!search and rescue
USB	universal!serial bus
USBL	ultrashort baseline
USBL	ultrashort-baseline
USC	University of Southern California
USV	unmanned!surface vehicle
UTC	universal coordinated time
UUB	uniform ultimate boundedness
UUV	unmanned underwater vehicle
UV	ultraviolet
UVMS	underwater vehicle!manipulator system
UWB	ultrawide band
UXO	unexploded ordnance

V

V2V	vehicle-to-vehicle
VAS	visual!analog scale
VCR	video!cassette recorder
vdW	van der Waals
VE	virtual environment
VFH	vector field histogram
VHF	very high frequency
VI	value iteration
VIA	variable impedance actuator
VIP	ventral intraparietal
VM	virtual!manipulator
VME	Versa Module Europa
VO	virtual object
VO	velocity obstacle
VOC	visual object class
VOR	vestibular-ocular reflex
VR	variable reluctance
VRML	virtual reality modeling language
VS	visual servo
VS-Joint	variable stiffness joint
VSA	variable stiffness actuator
VTOL	vertical take-off and landing

W

W3C	WWW consortium
WAAS	wide-area augmentation system
WABIAN	Waseda bipedal humanoid
WABOT	Waseda robot
WAM	whole-arm manipulator
WAN	wide-area network
WASP	wireless!ad-hoc system for positioning
WAVE	wireless!access in vehicular environments
WCF	worst-case factor
WCR	worst-case range
WDVI	weighted!difference vegetation index
WG	world!graph
WGS	World Geodetic System
WHOI	Woods Hole Oceanographic Institution
WML	wireless markup language
WMR	wheeled mobile robot
WSN	wireless!sensor network
WTA	winner-take-all
WTC	World Trade Center
WWW	world wide web

X

XCOM	extrapolated center of mass
XHTML	extensible hyper text markup language
XML	extensible markup language
xUCE	urban!challenge event

Y

YARP	yet another robot platform

Z

ZMP	zero moment point
ZOH	zero order hold
ZP	zona pellucida

Multimedia Contents



1. Robotics and the Handbook

Bruno Siciliano, Oussama Khatib

Robots! Robots on Mars and in oceans, in hospitals and homes, in factories and schools; robots fighting fires, making goods and products, saving time and lives. Robots today are making a considerable impact on many aspects of modern life, from industrial manufacturing to healthcare, transportation, and exploration of the deep space and sea. Tomorrow, robots will be as pervasive and personal as today's personal computers. This chapter retraces the evolution of this fascinating field from the ancient to the modern times through a number of milestones: from the first automated mechanical artifact (1400 BC) through the establishment of the robot concept in the 1920s, the realization of the first industrial robots in the 1960s, the definition of robotics science and the birth of an active research community in the 1980s, and the expansion towards the challenges of the human world of the twenty-first century. Robotics in its long journey has inspired this handbook which is organized in three layers: the foundations of robotics science; the consolidated methodologies and technologies of robot design, sensing and perception, manipulation and interfaces, mobile and distributed robotics; the advanced applications of field and service robotics, as well as of human-centered and life-like robotics.

1.1 **A Brief History of Robotics** 1
1.2 **The Robotics Community** 3
1.3 **This Handbook** 4
Video-References 5

1.1 A Brief History of Robotics

The dream to create machines that are skilled and intelligent has been part of humanity from the beginning of time. This dream is now becoming part of our world's striking reality. Since the early civilizations, one of man's greatest ambitions has been to create artifacts in their image. The legend of the Titan Prometheus, who molded humankind from clay, or that of the giant Talus, the bronze slave forged by Hephaestus (3500 BC), testify to this quest in Greek mythology. The Egyptians' oracle statues hiding priests inside (2500 BC) were perhaps the precursor of our modern thinking machines. The clepsydra water clock introduced by the Babylonians (1400 BC) was one of the first automated mechanical artifacts. In the following centuries, human creativity has given rise to a host of devices such as the automaton theatre of Hero of Alexandria (100 AD), the hydro-powered water-raising and humanoid machines of Al-Jazari (1200), and Leonardo da Vinci's numerous ingenious designs (1500). The development of automata continued to flourish in the eighteen century both in Europe and Asia, with creations such as Jacquet-Droz's family of androids (drawer, musician and writer) and the *karakuri-ningyo* mechanical dolls (tea server and archer).

The robot *concept* was clearly established by those many creative historical realizations. Nonetheless, the emergence of the *physical* robot had to await the advent of its underlying technologies during the course of the twentieth century. In 1920, the term robot – derived from *robota* which means subordinate labour in Slav languages – was first introduced by the Czech playwright Karel Čapek in his play *Rossum's Universal Robots* (R.U.R.). In 1940, the ethics of the interaction

between robots and humans was envisioned to be governed by the well-known three fundamental laws of Isaac Asimov, the Russian science-fiction writer in his novel *Runaround*.

The middle of the twentieth century brought the first explorations of the connection between human intelligence and machines, marking the beginning of an era of fertile research in the field of artificial intelligence (AI). Around that time, the first robots were realized. They benefited from advances in the different technologies of mechanics, controls, computers and electronics. As always, new designs motivate new research and discoveries, which, in turn, lead to enhanced solutions and thus to novel concepts. This virtuous circle over time produced that knowledge and understanding which gave birth to the field of *robotics*, properly referred to as: the science and technology of robots.

The early robots built in the 1960s stemmed from the confluence of two technologies: numerical control machines for precise manufacturing, and teleoperators for remote radioactive material handling. These master-slave arms were designed to duplicate one-to-one the mechanics of the human arm, and had rudimental control and little perception about the environment. Then, during the mid-to-late twentieth century, the development of integrated circuits, digital computers and miniaturized components enabled computer-controlled robots to be designed and programmed. These robots, termed industrial robots, became essential components in the automation of flexible manufacturing systems in the late 1970s. Further to their wide application in the automotive industry, industrial robots were successfully employed in general industry, such as the metal products, the chemical, the electronics and the food industries. More recently, robots have found new applications outside the factories, in areas such as cleaning, search and rescue, underwater, space, and medical applications.

In the 1980s robotics was defined as the science which studies the *intelligent connection between perception and action*. With reference to this definition, the action of a robotic system is entrusted to a *locomotion* apparatus to move in the environment (wheels, crawlers, legs, propellers) and/or to a *manipulation* apparatus to operate on objects present in the environment (arms, end effectors, artificial hands), where suitable *actuators* animate the mechanical components of the robot. The perception is extracted from the *sensors* providing information on the state of the robot (position and speed) and its surrounding environment (force and tactile, range and vision). The intelligent connection is entrusted to a *programming, planning and control* architecture which relies on the perception and available *models* of the robot and environment and exploits learning and skill acquisition.

In the 1990s research was boosted by the need to resort to robots to address human safety in hazardous environments (*field robotics*), or to enhance the human operator ability and reduce his/her fatigue (*human augmentation*), or else by the desire to develop products with wide potential markets aimed at improving the quality of life (*service robotics*). A common denominator of such application scenarios was the need to operate in a scarcely structured environment which ultimately requires increased abilities and a higher degree of *autonomy*.

The first video attached to this chapter *Robots – A 50 Year Journey* by Oussama Khatib (2000) shows the development of robotics through the first five decades (VIDEO 805).

By the dawn of the new millennium, robotics has undergone a major transformation in scope and dimensions. This expansion has been brought about by the maturity of the field and the advances in its related technologies. From a largely dominant industrial focus, robotics has been rapidly expanding into the challenges of the human world (*human-centered and life-like robotics*). The new generation of robots is expected to safely and dependably co-habitat with humans in homes, workplaces, and communities, providing support in services, entertainment, education, healthcare, manufacturing, and assistance.

Beyond its impact on physical robots, the body of knowledge robotics has produced is revealing a much wider range of applications reaching across diverse research areas and scientific disciplines, such as: biomechanics, haptics, neurosciences, virtual simulation, animation, surgery, and sensor networks among others. In return, the challenges of the new emerging areas are proving an abundant source of stimulation and insights for the field of robotics. It is indeed at the intersection of disciplines that the most striking advances are expected to happen.

The second video attached to this chapter *Robots – The Journey Continues* by Bruno Siciliano, Oussama Khatib and Torsten Kröger (2015) shows the intensive and vibrating evolution of robotics through the last fifteen years (VIDEO 812).

1.2 The Robotics Community

The dissemination of research results and findings in archival publications and conference presentations has played an important role in the advancement of robotics in the past three decades. The extent of scientific activities in robotics has led to the establishment of professional societies and research networks devoted to the field, which have ultimately contributed to the building of an international robotics community.

In the early 1980s, robotics was entering a new era in its development following the earlier successes in industrial applications. With much energy and excitement, the pioneers of robotics launched a number of initiatives, which came to shape what became our field. The first robotics textbook by Richard Paul *Robot Manipulators: Mathematics, Programming, and Control* appeared in 1981 (MIT Press). A year later, Mike Brady, Richard Paul, and their colleagues established the first journal entirely devoted to robotics, the International Journal of Robotics Research (IJRR), which was followed by the organization in 1983 of the first International Symposium of Robotics Research (ISRR). The intent was to foster the notion of robotics as a distinct scientific field and to create a broad community of robotics. It was in the pursuit of these aims that the International Foundation of Robotics Research (IFRR) was later formally founded in 1986 with the aim of promoting the development of robotics as a science, establishing the theoretical foundations and technology basis for its ever-expanding applications, with emphasis on its potential role to benefit humans.

The year 1984 brought a host of other major developments in robotics with the establishment of the IEEE Robotics and Automation Council (RAC), led by George Saridis, the organization of the first IEEE Conference on Robotics (John Jarvis and Richard Paul) – which became the IEEE International Conference on Robotics and Automation (ICRA) the year after, the launching of the IEEE Journal of Robotics and Automation (George Bekey), and RAC Newsletter (Wesley Snyder). The transformation of the council into the IEEE Robotics and Automation Society (RAS) was formally accomplished in 1989. With the birth of RAS, the IEEE Journal of Robotics and Automation became the IEEE Transactions on Robotics and Automation – which split into the IEEE Transactions on Robotics and the IEEE Transactions on Automation Science and Engineering later in 2004; then, the RAS Newsletter became the IEEE Robotics and Automation Magazine in 1994.

Over the following years, a number of other activities were launched. New conferences such as IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 1983), International Conference on Advanced Robotics (ICAR 1983), International Symposium on Experimental Robotics (ISER 1989), Robotics: Science and Systems (RSS 2005), as well as new journals such as Robotica (1983), Journal of Robotic Systems (1984, subsequently Journal of Field Robotics 2006), Robotics and Autonomous Systems (1985), Advanced Robotics (1986), Journal of Intelligent and Robotic Systems (1988), Autonomous Robots (1994), contributed to a further substantial growth of the robotics community.

The year 1991 marked an important novelty into the ICRA flagship conference series with the creation of video proceedings, introduced earlier by Oussama Khatib and Vincent Hayward in ISER 1989. Since then video proceedings attained an important role for dissemination of research work in our robotics community. This media has become an integral part of most archival publications in recent years.

On the other hand, the introduction of graduate programs in robotics in many academic institutions around the world in the last two decades is a clear illustration of the level of maturity reached by robotics as a scientific field. Further to Paul's seminal book, other textbooks were published between the middle 1980s and middle 1990s, such as *Introduction to Robotics: Mechanics and Control* by John Craig (1985), *Robot Analysis and Control* by Harry Asada and Jean-Jacques Slotine (1986), *Robotics: Control, Sensing, Vision, and Intelligence* by King Sun Fu, Rafael González and George Lee (1987), *Robot Dynamics and Control* by Mark Spong and M. Vidyasagar (1989), *Foundation of Robotics* by Tsuneo Yoshikawa (1990), *A Mathematical Introduction to Robotic Manipulation* by Richard Murray, Zexiang Li and Shankar Sastry (1994), and *Modelling and Control of Robot Manipulators* by Lorenzo Sciavicco and Bruno Siciliano (1995).

Engagement with students is essential for the future of the robotics field. Through a joint initiative between IEEE RAS and IFRR, the School of Robotics Science was founded in 2004. Summer schools have rapidly expanded in scope and frequency, covering a wide range of robotics research areas. Such programs are providing students and young researchers with an educational opportunity of the highest scientific level, enabling them to feed and grow their potential to become the top robotics scientists in our community.

Both IEEE RAS and IFRR have constantly promoted research, education, dissemination, and industrial realizations through their own forums and through collaborations with many other organizations

in the robotics community world-wide, including European Robotics Research Network (EURON) – which has recently merged into the European Association of Robotics (euRobotics), International Federation of Robotics (IFR), and Robotics Society of Japan (RSJ).

As our field continues to expand and embrace the challenges of operation in the human world, robotics is revealing a much wider range of applications and involving diverse scientific disciplines such as biomechanics, neuroscience, haptics, animation, surgery, and sensor networks among others. Today, new communities of researchers and developers are forming, with growing connections to the core of robotics research. A strategic goal for our community is one of outreach and scientific cooperation across all regions and organizations. Our common endeavor as roboticists to reach this goal certainly holds the promise of exciting advances toward new frontiers.

1.3 This Handbook

The intensive stream of robotics research documented in the literature culminates into this unique reference, which aims at collecting in one self-contained volume the most significant achievements of our international robotics community. The *Springer Handbook of Robotics* presents a full coverage of the field from its foundations, through the research areas, up to the new emerging applications of robotics. Accordingly, the material is organized in three logical layers reflecting the historical development of the field, as illustrated in Fig. 1.1.

The foundations of robotics science, laid down in the first layer (Part A with its 14 chapters), address the theory of robot mechanics, sensing, planning, and control. The consolidated methodologies and technologies of robot design (Part B with its 12 chapters), sensing and perception (Part C with its 8 chapters), manipulation and interfaces (Part D with its 9 chapters) and mobile and distributed robotics (Part E with its 9 chapters) are presented in the second layer. The third layer is devoted to advanced applications, such as in field and service robotics (Part F with its 13 chapters) and human-centered and life-like robotics (Part G with its 14 chapters).

Part A presents the fundamental principles and methods that are used to model, design, and control a robotic system. All of the foundational topics are included in this part: kinematics, dynamics, mechanisms and actuation, sensing and estimation, model identification, motion planning, motion control, force control, redundant robots, robots with flexible elements, robotic systems architectures and programming, behaviour-based systems, AI reasoning methods for robotics, robot learning. A chapter is devoted to each of these topics. The topics are expanded and applied to specific robotic structures and systems in subsequent parts.

Part B is concerned with the design of various robotic systems. Some of the obvious mechanical structures that come to mind are arms, legs, wheels and hands; to this list can be added robots in the air

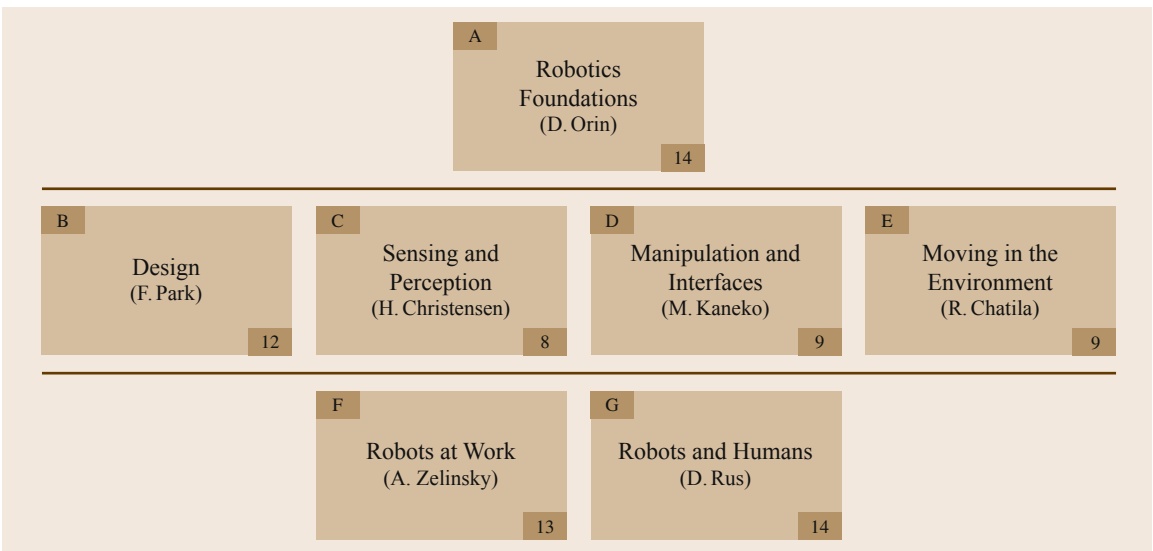


Fig. 1.1 Organization of the handbook

and subsea, as well as robot structures at the micro and nano scales. With a separate chapter devoted to design and evaluation performance, the chapters in this part successively examine limbed systems, parallel mechanisms, robot hands, snake-like and continuum robots, novel actuators for soft robotics, modular robots, biomimetic robots, wheeled robots, underwater robots, flying robots, and micro/nano robots.

Part C covers different sensory modalities and integration of sensor data across space and time to generate models of robots and the external environment. Robotics is the intelligent coupling of perception and action and as such Part C complements Part B to build systems. This part of the handbook covers sensing across contact, proprioception and exteroception. The main sensing modalities such as force and tactile sensing, inertial sensing, global positioning system (GPS) and odometry, sonar sensing, range sensing, three-dimensional (3-D) vision for navigation and grasping visual object class recognition, and visual servoing are presented. Both basic sensor models, sensor data processing and associated representations are covered. Finally, a chapter on multisensor data fusion introduces the mathematical tools needed for integration of sensor information across space and time.

Part D is concerned with interaction between robots and objects, or between humans and robots. Manipulation is supposed to handle an object through direct contact by arms or fingers or just a pusher, while interfaces are supposed to make either direct or indirect interaction between humans and robots. For enhancing dexterity in robot manipulation, motion for manipulation tasks, contact modelling and manipulation, grasping, cooperative manipulation, mobility and manipulation, active manipulation for perception are addressed in the first half of this part. For achieving a skilful manipulation or power increase in a human–robot system, haptics, telerobotics, and networked robots are discussed in the second half of Part D.

Part E covers a wide span of topics concerning the motion of robots in the environment. At first this part addresses world modelling, simultaneous localization and mapping, motion planning and obstacle avoidance. Then the modelling and control issues of legged robots, wheeled mobile robots, robots on rough terrain, underwater robots, and aerial robots are dealt with. This part completes Part A on foundations in the context of mobile robotics, and given the role of perception, is closely related to Part C on sensing. In addition, multiple mobile robot systems are discussed.

Part F covers topics related to creating field and service application-based robots that operate in all types of environments. This includes applications ranging from industrial robots to robotics competitions and challenges, through a diverse array of robotics applications: space, agriculture and forestry, construction, hazardous applications, mining, disaster scenarios, intelligent vehicles, medical and computer-integrated surgery, rehabilitation and health-care, domestic environments. This part of the handbook, which draws on Parts A, B, C, D and E, describes how robots can be put to work.

Part G covers topics related to creating robots that operate in human-centered environments. These include humanoids, human motion reconstruction, physical human–robot interaction, human robot augmentation, cognitive human–robot interaction, social robotics, socially assistive robotics, learning from humans, biologically-inspired robotics, evolutionary robotics, neurobotics, and perceptual robotics. The part concludes with the use of robotics for education, and the socio-ethical implications of robots.

Each part is accompanied by a video which illustrates the main concepts and the underlying challenges in common to the chapters in the part, while topical videos related to the specific contents of each chapter can be accessed through the web site of the multimedia contents associated with the handbook http://handbookofrobotics.org/.

The handbook has been conceived to provide a valuable resource not only for robotics experts, but also for newcomers to this expanding field, e.g., engineers, medical doctors, computer scientists, and designers. In particular, it is important to underline the tutorial value of Part A to graduate students and post-docs, the research value of Parts B to E for a wider coverage of research in robotics, and the added value of Parts F and G to engineers and scientists interested in new applications.

The contents of the various chapters have been inspired by a classic cut, i. e., avoiding the inclusion of on-going or not well-established methods. An objective perspective has been taken, while covering multiple approaches, with the goal of ensuring a high archival value to the handbook. Each chapter is preceded by a short summary, and an introductory section providing the state of the art in the area. The core sections are developed at a tutorial level. Lengthy mathematical derivations have been avoided whenever possible, while the equations, tables, and algorithms are illustrated in ready-to-use form. The final section of the chapter provides conclusions and topics for further reading. From the foundations to the social and ethical implications of robotics, the eighty chapters of the handbook provide a comprehensive collection of five decades of progress in robotics. This sum is a testimony to the level of accomplishments in our field, and a premise of further advances towards new frontiers of robotics.

Video-References

VIDEO 805 Robots – A 50 year journey
available from http://handbookofrobotics.org/view-chapter/01/videodetails/805

VIDEO 812 Robots – The journey continues
available from http://handbookofrobotics.org/view-chapter/01/videodetails/812

Multimedia Contents



Part A Robotics Foundations

Ed. by David E. Orin

2 Kinematics
Kenneth J. Waldron, Ultimo, Australia
James Schmiedeler, Notre Dame, USA

3 Dynamics
Roy Featherstone, Canberra, Australia
David E. Orin, Columbus, USA

4 Mechanism and Actuation
Victor Scheinman, Stanford, USA
J. Michael McCarthy, Irvine, USA
Jae-Bok Song, Seoul, Korea

5 Sensing and Estimation
Henrik I. Christensen, Atlanta, USA
Gregory D. Hager, Baltimore, USA

6 Model Identification
John Hollerbach, Salt Lake City, USA
Wisama Khalil, Nantes, France
Maxime Gautier, Nantes, France

7 Motion Planning
Lydia E. Kavraki, Houston, USA
Steven M. LaValle, Urbana, USA

8 Motion Control
Wan Kyun Chung, Pohang, Korea
Li-Chen Fu, Taipei, Taiwan
Torsten Kröger, Mountain View, USA

9 Force Control
Luigi Villani, Naples, Italy
Joris De Schutter, Leuven-Heverlee, Belgium

10 Redundant Robots
Stefano Chiaverini, Cassino, Italy
Giuseppe Oriolo, Rome, Italy
Anthony A. Maciejewski, Fort Collins, USA

11 Robots with Flexible Elements
Alessandro De Luca, Rome, Italy
Wayne J. Book, Atlanta, USA

12 Robotic Systems Architectures and Programming
David Kortenkamp, Houston, USA
Reid Simmons, Pittsburgh, USA
Davide Brugali, Dalmine, Italy

13 Behavior-Based Systems
François Michaud, Sherbrooke, Canada
Monica Nicolescu, Reno, USA

14 AI Reasoning Methods for Robotics
Michael Beetz, Bremen, Germany
Raja Chatila, Paris, France
Joachim Hertzberg, Osnabrück, Germany
Federico Pecora, Örebro, Sweden

15 Robot Learning
Jan Peters, Darmstadt, Germany
Daniel D. Lee, Philadelphia, USA
Jens Kober, Delft, The Netherlands
Duy Nguyen-Tuong, Stuttgart, Germany
J. Andrew Bagnell, Pittsburgh, USA
Stefan Schaal, Los Angeles, USA

The chapters contained in Part A, Robotics Foundations, present the fundamental principles and methods that are used to develop a robotic system. In order to perform the tasks that are envisioned for robots, many challenging problems have been uncovered in kinematics, dynamics, design, actuation, sensing, modeling, motion planning, control, programming, decision-making, task planning, and learning. Robots with redundant kinematic degrees of freedom or with flexible elements add to the complexity of these systems. The chapters in this part address the basic issues in each of these areas.

Some of the basic problems in robotics are outlined as follows. Robots often consist of a large number of degrees of freedom so that they can provide the rich set of three-dimensional (3-D) motions that may be required for a range of tasks. The kinematic and dynamic relationships between the joint actuators' motion and torques, and the desired motion and force for a task can be very complex. The design of the link and joint structures, as well as the actuation, to achieve the desired performance is also challenging.

The robot is a nonlinear, coupled system which is difficult to model and control because of its complex dynamics. Kinematic redundancy and flexible elements in the robots increase this complexity. The problem is exacerbated when the environment is unstructured, and often sophisticated sensing and estimation techniques are required.

In addition to control of the motion, control of the interaction forces between the robot and environment is needed when manipulating objects or interacting with humans. A fundamental robotics task is to plan collision-free motion for complex bodies from a start to a goal position among a collection of obstacles, and this can become an intractable computational problem.

In order to achieve some of the intelligence ascribed to humans, robots will need to be equipped with sophisticated action planners that employ symbolic reasoning to move in dynamic, partially known environments. In other scenarios, robots need to execute behaviors that take advantage of dynamic interactions with the environment rather than rely solely on explicit reasoning and planning. Robot software architectures also have special needs because of all of these requirements. Robot learning will be necessary to generate actions and control for ever-changing task requirements and environments, in order to achieve the level of autonomy envisioned.

While the basic issues outlined in the previous paragraphs are addressed in this part, more depth can be found in other parts of the handbook. The kinematics, dynamics, mechanical design, and control principles and methods introduced in this part can be applied to robotic structures made up of arms, hands, and legs (Part B) as well as manipulators (Part D), wheeled and other mobile robots (Part E), and field and service robots (Part F). Force control is especially important for manipulators and their interfaces (Part D). The basic sensing and estimation techniques presented here are expanded and applied to specific sensing modalities in Part C. Motion planning is an important aspect of manipulation (Part D) and mobile robots moving in the environment (Part E). Robotic systems architectures, behavior-based systems, artificial intelligence (AI) reasoning methods, and robot learning are particularly important in mobile robots moving in the environment (Part E) and robots interacting with humans (Part G).

With this overview of Part A, we now provide a brief synopsis of each chapter:

Chapter 2, Kinematics, provides a number of representations and conventions to describe the motion of the bodies in a robotic mechanism. These include rotation matrices, Euler angles, quaternions, homogeneous transformations, screw transformations, matrix exponential parameterization, and Plücker coordinates. Representations of the kinematics of all common joint types are provided, along with a modified form of the Denavit–Hartenberg convention. These representational tools are applied to compute the workspace, the forward and inverse kinematics, the forward and inverse instantaneous kinematics, Jacobian, and the static wrench transmission.

Chapter 3, Dynamics, presents the dynamic equations of motion which provide the relationships between actuation and contact forces acting on robot mechanisms, and the acceleration and motion trajectories that result. Efficient algorithms are provided for important dynamics computations which include inverse dynamics, forward dynamics, the joint-space inertia matrix, and the operational-space inertia matrix. The algorithms may be applied to fixed-base robots, mobile robots, and parallel robot mechanisms. Compact formulation of the algorithms results from using six-dimensional (6-D) spatial notation to describe rigid-body velocity, acceleration, inertia, etc.

Chapter 4, Mechanisms and Actuation, focuses on the principles that guide the design and construction of robotic systems. The kinematic equations and Jacobian are used to characterize the work envelope and mechanical advantage, and guide the selection of the robot's size and joint arrangement. The design of both serial and parallel robots is addressed. Practical consideration

is given to the design of the link and joint structures along with selection of the actuators and transmission drives to power the movement. Robot performance in terms of speed, acceleration, repeatability, and other measures is also addressed.

Chapter 5, Sensing and Estimation, provides a brief overview of common sensing methods and estimation techniques that have found broad applicability in robotics. These provide information about the state of the environment and robot system. The presentation is structured according to a perception process model that includes sensing, feature extraction, data association, parameter estimation and model integration. Several common sensing modalities are introduced and characterized. Methods for estimation in linear and nonlinear systems are discussed, including statistical estimation, the Kalman filter, and sample-based methods. The several common representations for estimation are also introduced.

Chapter 6, Model Identification, discusses methods for determining the kinematic and inertial parameters of robot manipulators. For kinematic calibration, the primary aim is to identify the geometric Denavit–Hartenberg parameters or equivalent, typically by sensing a combination of joint and endpoint positions. Inertial parameters in turn are estimated through the execution of a trajectory while additionally sensing one or more components of joint forces or torques. The chapter is organized such that both kinematic and inertial parameter identification are cast into a common framework of least-squares parameter estimation: common features relating to the identifiability of the parameters, adequacy of the measurement sets, and numerical robustness are identified and emphasized for both the kinematic and inertial parameters.

Chapter 7, Motion Planning, addresses the fundamental robotics task to plan collision-free motion for complex bodies from a start to a goal position among a collection of obstacles. The basic geometric path planning problem (Piano Mover's problem) is introduced, but the focus of the chapter is on sampling-based planning methods because of their generally wider applicability. Planning with differential constraints is considered and is important for wheeled mobile robots. Extensions and variations to the basic motion planning problem, as well as advanced issues, are discussed at the end of the chapter.

Chapter 8, Motion Control, focuses on the control of the motion of a rigid manipulator. The main challenges addressed are the complexity of the nonlinear, coupled dynamics and the model structural uncertainties. The chapter discusses topics ranging from independent-joint control and PID (proportional–integral–derivative) control to computed-torque control to manage the complex dynamics. Operational space (or task space) control is presented, and this is needed for control of tasks such as those associated with the end-effector. Adaptive and robust control are discussed to address the problems related to the uncertainties in the system. Other topics presented include trajectory generation, digital implementation, and learning control.

Chapter 9, Force Control, focuses on the control of the interaction forces between a robotic system and its environment. The chapter groups interaction control in two categories: indirect and direct force control, which achieve the control without (indirect) or with (direct) explicit closure of a force feedback loop. Impedance control and hybrid force/motion control are examples of each, respectively. The fundamental problem of interaction tasks modeling is presented to provide the basis for the force control schemes.

Chapter 10, Redundant Robots, addresses the motion generation and control of manipulators with redundant kinematic degrees of freedom. Kinematic redundancy affords a robot with an increased level of dexterity that may be used to, e.g., avoid singularities, joint limits and workspace obstacles, and also to minimize joint torques, energy, or other suitable performance criteria. The chapter discusses inverse kinematic redundancy resolution schemes which are arranged in two main categories, namely those based on the optimization of suitable performance criteria and those relying on the augmentation of the task space. The use of kinematic redundancy for fault tolerance is also analyzed.

Chapter 11, Robots with Flexible Elements, addresses the dynamic modeling and control of robots with flexibility in the joints and links. Because the control methods developed to compensate for joint versus link flexibility are structurally different, the chapter is organized such that these two types of flexibility are examined independently. The methods, however, can be extended to the case when both joint and link flexibilities are present, possibly even dynamically interacting at the same time. The chapter also examines the typical sources of flexibility in industrial robots.

Chapter 12, Robotic Systems Architectures and Programming, presents the software architectures and supporting programming tools and environments that have been developed for robotic systems. Robot architectures have special needs because of the requirements of the robot to interact asynchronously, in real time, with an uncertain, often dynamic, environment. The chapter discusses the major types of architectural com-

ponents for a layered robot control architecture – behavioral control, executives, and task planners – along with the commonly used techniques for interconnecting these components.

Chapter 13, Behavior-Based Systems, describes a control methodology aimed at situated robots operating in unconstrained, challenging, and dynamic conditions in the real world. Distributed behaviors are used as the underlying building blocks, allowing behavior-based systems to take advantage of dynamic interactions with the environment rather than rely solely on explicit reasoning and planning. The focus of the chapter is to provide the basic principles of behavior-based systems and their use in autonomous control problems and applications. The chapter presents several different classes of learning methods, but in all cases behaviors are used as the underlying building blocks for the learning process.

Chapter 14, AI Reasoning Methods for Robotics, sketches the main robotics-relevant topics of symbol-based AI reasoning. Reasoning on a mobile robot is especially challenging because of its dynamic, partially known environment. This chapter describes basic methods of knowledge representation and inference, covering both logic- and probability-based approaches. Results for several types of practical, robotics-related reasoning tasks are given, with an emphasis on temporal and spatial reasoning. Plan-based robot control is described in some detail, as it is a particularly obvious form of employing reasoning to the end of improving robot action.

Chapter 15, Robot Learning, surveys work in machine learning for techniques that are used for learning control and behavior generation in robots. In the future, robots will no longer be used to only execute the same task thousands of times, but rather they will be faced with thousands of different tasks that rarely repeat in an ever-changing environment. Robot learning will be necessary to achieve the high degree of autonomy envisioned. This chapter focuses on the core robot learning approaches capable of learning action generation and control, and in particular, techniques for model learning and reinforcement learning.

Part A presents the fundamental principles and methods that are used to model, design, and control a robotic system. All of the foundational topics are included in this part: kinematics, dynamics, mechanisms and actuation, sensing and estimation, model identification, motion planning, motion control, force control, redundant robots, robots with flexible elements, robotic systems architectures and programming, behavior-based systems, AI reasoning methods for robotics, and robot learning. A chapter is devoted to each of these topics. The topics are expanded and applied to specific robotic structures and systems in subsequent parts.

2. Kinematics

Multimedia Contents

Kenneth J. Waldron, James Schmiedeler

Kinematics pertains to the motion of bodies in a robotic mechanism without regard to the forces/torques that cause the motion. Since robotic mechanisms are by their very essence designed for motion, kinematics is the most fundamental aspect of robot design, analysis, control, and simulation. The robotics community has focused on efficiently applying different representations of position and orientation and their derivatives with respect to time to solve foundational kinematics problems.

This chapter will present the most useful representations of the position and orientation of a body in space, the kinematics of the joints most commonly found in robotic mechanisms, and a convenient convention for representing the geometry of robotic mechanisms. These representational tools will be applied to compute the *workspace*, the *forward* and *inverse kinematics*, the *forward* and *inverse instantaneous kinematics*, and the *static wrench transmission* of a robotic mechanism. For brevity, the focus will be on algorithms applicable to open-chain mechanisms.

The goal of this chapter is to provide the reader with general tools in tabulated form and a broader overview of algorithms that can be applied together to solve kinematics problems pertaining to a particular robotic mechanism.

2.1 **Overview** 12
2.2 **Position and Orientation Representation** 12
2.2.1 Position and Displacement 12
2.2.2 Orientation and Rotation 13
2.2.3 Homogeneous Transformations 16
2.2.4 Screw Transformations 17
2.2.5 Matrix Exponential Parameterization 19
2.2.6 Plücker Coordinates 20
2.3 **Joint Kinematics** 21
2.3.1 Lower Pair Joints 21
2.3.2 Higher Pair Joints 24
2.3.3 Compound Joints 24
2.3.4 6-DOF Joint 24
2.3.5 Physical Realization 25
2.3.6 Holonomic and Nonholonomic Constraints 25
2.3.7 Generalized Coordinates 25
2.4 **Geometric Representation** 25
2.5 **Workspace** 27
2.6 **Forward Kinematics** 28
2.7 **Inverse Kinematics** 29
2.7.1 Closed-Form Solutions 29
2.7.2 Numerical Methods 30
2.8 **Forward Instantaneous Kinematics** 31
2.8.1 Jacobian 31
2.9 **Inverse Instantaneous Kinematics** 32
2.9.1 Inverse Jacobian 32
2.10 **Static Wrench Transmission** 33
2.11 **Conclusions and Further Reading** 33
References 33

2.1 Overview

Unless explicitly stated otherwise, robotic mechanisms are systems of rigid bodies connected by joints. The position and orientation of a rigid body in space are collectively termed the *pose*. Therefore, robot kinematics describes the pose, velocity, acceleration, and all higher-order derivatives of the pose of the bodies that comprise a mechanism. Since kinematics does not address the forces/torques that induce motion, this chapter focuses on describing pose and velocity. These descriptions are foundational elements of dynamics (Chap. 3), motion planning (Chap. 7), and motion control (Chap. 8) algorithms.

Among the many possible topologies in which systems of bodies can be connected, two of particular importance in robotics are serial chains and fully parallel mechanisms. A serial chain is a system of rigid bodies in which each member is connected to two others, except for the first and last members that are each connected to only one other member. A fully parallel mechanism is one in which there are two members that are connected together by multiple chains of other members and joints. In practice, each of these chains is often itself a serial chain. This chapter focuses almost exclusively on algorithms applicable to serial chains. Parallel mechanisms are dealt with in more detail in Chap. 18. Another important topology is the tree structure, which is similar to a serial chain in that it has no closed loops, but differs from a serial chain in that each member might have multiple members connected to it, forming multiple *branches*. A serial chain is actually just a special case of a tree structure with no branches. Tree structures are addressed in greater depth in Chap. 3.

2.2 Position and Orientation Representation

Spatial, rigid-body kinematics can be viewed as a comparative study of different ways of representing the pose of a body. Translations and rotations, referred to in combination as rigid-body displacements, are also expressed with these representations. No one approach is optimal for all purposes, but the advantages of each can be leveraged appropriately to facilitate the solution of different problems.

The minimum number of coordinates required to locate a body in Euclidean space is six. Many representations of spatial pose employ sets with superabundant coordinates in which auxiliary relationships exist among the coordinates. The number of independent auxiliary relationships is the difference between the number of coordinates in the set and six.

This chapter and those that follow it make frequent use of *coordinate frames* or simply *frames*. A coordinate frame i consists of an origin, denoted O_i, and a triad of mutually orthogonal basis vectors, denoted $(\hat{\boldsymbol{x}}_i\ \hat{\boldsymbol{y}}_i\ \hat{\boldsymbol{z}}_i)$, that are all fixed within a particular body. The pose of a body will always be expressed relative to some other body, so it can be expressed as the pose of one coordinate frame relative to another. Similarly, rigid-body displacements can be expressed as displacements between two coordinate frames, one of which may be referred to as *moving*, while the other may be referred to as *fixed*. This indicates that the observer is located in a stationary position within the fixed frame, not that there exists any absolutely fixed frame.

2.2.1 Position and Displacement

The position of the origin of coordinate frame i relative to coordinate frame j can be denoted by the 3×1 vector

$$ {}^j\boldsymbol{p}_i = \begin{pmatrix} {}^jp_i^x \\ {}^jp_i^y \\ {}^jp_i^z \end{pmatrix} . $$

The components of this vector are the Cartesian coordinates of O_i in the j frame, which are the projections of the vector ${}^j\boldsymbol{p}_i$ onto the corresponding axes. The vector components could also be expressed as the spherical or cylindrical coordinates of O_i in the j frame. Such representations have advantages for analysis of robotic mechanisms including spherical and cylindrical joints.

A translation is a displacement in which no point in the rigid body remains in its initial position and all straight lines in the rigid body remain parallel to their initial orientations. (The points and lines are not necessarily contained within the boundaries of the finite rigid body, but rather, any point or line in space can be taken to be rigidly fixed in a body.) The translation of a body in space can be represented by the combination of its positions prior to and following the translation. Conversely, the position of a body can be represented as a translation that takes the body from a starting position (in which the coordinate frame fixed to the body coincides with the fixed coordinate frame) to the current position (in which the two frames are not coincident).

Thus, any representation of position can be used to create a representation of displacement, and vice versa.

2.2.2 Orientation and Rotation

There is significantly greater breadth in the representation of orientation than in that of position. This section does not include an exhaustive summary, but focuses on the representations most commonly applied to robotic mechanisms.

A rotation is a displacement in which at least one point in the rigid body remains in its initial position and not all lines in the body remain parallel to their initial orientations. For example, a body in a circular orbit rotates about an axis through the center of its circular path, and every point on the axis of rotation is a point in the body that remains in its initial position. As in the case of position and translation, any representation of orientation can be used to create a representation of rotation, and vice versa.

Rotation Matrices

The orientation of coordinate frame i relative to coordinate frame j can be denoted by expressing the basis vectors $(\hat{\boldsymbol{x}}_i\,\hat{\boldsymbol{y}}_i\,\hat{\boldsymbol{z}}_i)$ in terms of the basis vectors $(\hat{\boldsymbol{x}}_j\,\hat{\boldsymbol{y}}_j\,\hat{\boldsymbol{z}}_j)$. This yields $({}^j\hat{\boldsymbol{x}}_i\,{}^j\hat{\boldsymbol{y}}_i\,{}^j\hat{\boldsymbol{z}}_i)$, which when written together as a 3×3 matrix is known as the rotation matrix. The components of ${}^j\mathbf{R}_i$ are the dot products of the basis vectors of the two coordinate frames.

$$
{}^j\mathbf{R}_i = \begin{pmatrix} \hat{\boldsymbol{x}}_i\cdot\hat{\boldsymbol{x}}_j & \hat{\boldsymbol{y}}_i\cdot\hat{\boldsymbol{x}}_j & \hat{\boldsymbol{z}}_i\cdot\hat{\boldsymbol{x}}_j \\ \hat{\boldsymbol{x}}_i\cdot\hat{\boldsymbol{y}}_j & \hat{\boldsymbol{y}}_i\cdot\hat{\boldsymbol{y}}_j & \hat{\boldsymbol{z}}_i\cdot\hat{\boldsymbol{y}}_j \\ \hat{\boldsymbol{x}}_i\cdot\hat{\boldsymbol{z}}_j & \hat{\boldsymbol{y}}_i\cdot\hat{\boldsymbol{z}}_j & \hat{\boldsymbol{z}}_i\cdot\hat{\boldsymbol{z}}_j \end{pmatrix} . \tag{2.1}
$$

Because the basis vectors are unit vectors and the dot product of any two unit vectors is the cosine of the angle between them, the components are commonly referred to as direction cosines.

An elementary rotation of frame i about the $\hat{\boldsymbol{z}}_j$ axis through an angle θ is

$$
\mathbf{R}_Z(\theta) = \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} , \tag{2.2}
$$

while the same rotation about the $\hat{\boldsymbol{y}}_j$ axis is

$$
\mathbf{R}_Y(\theta) = \begin{pmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{pmatrix} , \tag{2.3}
$$

and about the $\hat{\boldsymbol{x}}_j$ axis is

$$
\mathbf{R}_X(\theta) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{pmatrix} . \tag{2.4}
$$

Rotation matrices are combined through simple matrix multiplication such that the orientation of frame i relative to frame k can be expressed as

$$
{}^k\mathbf{R}_i = {}^k\mathbf{R}_j\,{}^j\mathbf{R}_i \, .
$$

The rotation matrix ${}^j\mathbf{R}_i$ contains nine elements, while only three parameters are required to define the orientation of a body in space. Therefore, six auxiliary relationships exist among the elements of the matrix. Because the basis vectors of coordinate frame i are mutually orthonormal, as are the basis vectors of coordinate frame j, the columns of ${}^j\mathbf{R}_i$ formed from the dot products of these vectors are also mutually orthonormal. A matrix composed of mutually orthonormal vectors is known as an orthogonal matrix and has the property that its inverse is simply its transpose. This property provides the six auxiliary relationships. Three require the column vectors to have unit length, and three require the column vectors to be mutually orthogonal. Alternatively, the orthogonality of the rotation matrix can be seen by considering the frames in reverse order. The orientation of coordinate frame j relative to coordinate frame i is the rotation matrix ${}^i\mathbf{R}_j$ whose rows are clearly the columns of the matrix ${}^j\mathbf{R}_i$.

In summary, ${}^j\mathbf{R}_i$ is the rotation matrix that transforms a vector expressed in coordinate frame i to a vector expressed in coordinate frame j. It provides a representation of the orientation of frame i relative to j and thus, can be a representation of rotation from frame i to frame j. Table 2.1 lists the equivalent rotation matrices for the other representations of orientation listed in this section. Table 2.2 contains the conversions from a known rotation matrix to these other representations.

Euler Angles

For a minimal representation, the orientation of coordinate frame i relative to coordinate frame j can be denoted as a vector of three angles $(\alpha,\beta,\gamma)^T$. These angles are known as Euler angles when each represents a rotation about an axis of a moving coordinate frame. In this way, the location of the axis of each successive rotation depends upon the preceding rotation(s), so the order of the rotations must accompany the three angles to define the orientation. For example, the symbols $(\alpha,\beta,\gamma)^T$ are used throughout this handbook to indicate Z-Y-X Euler angles. Taking the moving frame i and the fixed frame j to be initially coincident, α is the rotation about the $\hat{\boldsymbol{z}}$ axis of frame i, β is the rotation about the rotated $\hat{\boldsymbol{y}}$ axis of frame i, and finally, γ is the rotation about the twice rotated $\hat{\boldsymbol{x}}$ axis of frame i. The equivalent rotation matrix ${}^j\mathbf{R}_i$ is given in Table 2.1. Z-Y-Z and Z-X-Z Euler angles are other commonly used con-

Table 2.1 Equivalent rotation matrices for various representations of orientation, with abbreviations $c_\theta := \cos\theta$, $s_\theta := \sin\theta$, and $v_\theta := 1-\cos\theta$

Z-Y-X Euler angles $(\alpha,\beta,\gamma)^{\mathrm{T}}$	${}^{j}\mathbf{R}_i = \begin{pmatrix} c_\alpha c_\beta & c_\alpha s_\beta s_\gamma - s_\alpha c_\gamma & c_\alpha s_\beta c_\gamma + s_\alpha s_\gamma \\ s_\alpha c_\beta & s_\alpha s_\beta s_\gamma + c_\alpha c_\gamma & s_\alpha s_\beta c_\gamma - c_\alpha s_\gamma \\ -s_\beta & c_\beta s_\gamma & c_\beta c_\gamma \end{pmatrix}$
X-Y-Z fixed angles $(\psi,\theta,\phi)^{\mathrm{T}}$	${}^{j}\mathbf{R}_i = \begin{pmatrix} c_\phi c_\theta & c_\phi s_\theta s_\psi - s_\phi c_\psi & c_\phi s_\theta c_\psi + s_\phi s_\psi \\ s_\phi c_\theta & s_\phi s_\theta s_\psi + c_\phi c_\psi & s_\phi s_\theta c_\psi - c_\phi s_\psi \\ -s_\theta & c_\theta s_\psi & c_\theta c_\psi \end{pmatrix}$
Angle-axis $\theta\hat{\boldsymbol{w}}$	${}^{j}\mathbf{R}_i = \begin{pmatrix} w_x^2 v_\theta + c_\theta & w_x w_y v_\theta - w_z s_\theta & w_x w_z v_\theta + w_y s_\theta \\ w_x w_y v_\theta + w_z s_\theta & w_y^2 v_\theta + c_\theta & w_y w_z v_\theta - w_x s_\theta \\ w_x w_z v_\theta - w_y s_\theta & w_y w_z v_\theta + w_x s_\theta & w_z^2 v_\theta + c_\theta \end{pmatrix}$
Unit quaternions $(\epsilon_0\ \epsilon_1\ \epsilon_2\ \epsilon_3)^{\mathrm{T}}$	${}^{j}\mathbf{R}_i = \begin{pmatrix} 1-2\left(\epsilon_2^2+\epsilon_3^2\right) & 2\left(\epsilon_1\epsilon_2-\epsilon_0\epsilon_3\right) & 2\left(\epsilon_1\epsilon_3+\epsilon_0\epsilon_2\right) \\ 2\left(\epsilon_1\epsilon_2+\epsilon_0\epsilon_3\right) & 1-2\left(\epsilon_1^2+\epsilon_3^2\right) & 2\left(\epsilon_2\epsilon_3-\epsilon_0\epsilon_1\right) \\ 2\left(\epsilon_1\epsilon_3-\epsilon_0\epsilon_2\right) & 2\left(\epsilon_2\epsilon_3+\epsilon_0\epsilon_1\right) & 1-2\left(\epsilon_1^2+\epsilon_2^2\right) \end{pmatrix}$

ventions from among the 12 different possible orders of rotations.

Regardless of the order of rotations, an Euler angle representation of orientation always exhibits a singularity when the first and last rotations both occur about the same axis. This can be readily seen in the second block of Table 2.2 wherein the angles α and γ are undefined when $\beta = \pm 90°$. (For Z-Y-Z and Z-X-Z Euler angles, the singularity occurs when the second rotation is 0° or 180°.) This creates a problem in relating the angular velocity vector of a body to the time derivatives of Euler angles, which somewhat limits their usefulness in modeling robotic systems. This velocity relationship for Z-Y-X Euler angles is

$$\begin{pmatrix} \dot{\alpha} \\ \dot{\beta} \\ \dot{\gamma} \end{pmatrix} = \frac{1}{\cos\beta} \begin{pmatrix} 0 & \sin\gamma & \cos\gamma \\ 0 & \cos\gamma\cos\beta & -\sin\gamma\cos\beta \\ \cos\beta & \sin\gamma\sin\beta & \cos\gamma\sin\beta \end{pmatrix} \begin{pmatrix} \omega_x \\ \omega_y \\ \omega_z \end{pmatrix}, \tag{2.5}$$

where $(\omega_x,\omega_y,\omega_z)^{\mathrm{T}} = {}^{i}\boldsymbol{\omega}_i$ is given in moving frame i. In some circumstances, the inverse of this relationship may be required.

$$\begin{pmatrix} \omega_x \\ \omega_y \\ \omega_z \end{pmatrix} = \begin{pmatrix} -\sin\beta & 0 & 1 \\ \cos\beta\sin\gamma & \cos\gamma & 0 \\ \cos\beta\cos\gamma & -\sin\gamma & 0 \end{pmatrix} \begin{pmatrix} \dot{\alpha} \\ \dot{\beta} \\ \dot{\gamma} \end{pmatrix}. \tag{2.6}$$

Table 2.2 Conversions from a rotation matrix to various representations of orientation

Rotation matrix:

$${}^{j}\mathbf{R}_i = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix}$$

Z-Y-X Euler angles $(\alpha,\beta,\gamma)^{\mathrm{T}}$:

$$\beta = \mathrm{Atan2}\left(-r_{31}, \sqrt{r_{11}^2+r_{21}^2}\right)$$
$$\alpha = \mathrm{Atan2}\left(\frac{r_{21}}{\cos\beta}, \frac{r_{11}}{\cos\beta}\right)$$
$$\gamma = \mathrm{Atan2}\left(\frac{r_{32}}{\cos\beta}, \frac{r_{33}}{\cos\beta}\right)$$

X-Y-Z fixed angles $(\psi,\theta,\phi)^{\mathrm{T}}$:

$$\theta = \mathrm{Atan2}\left(-r_{31}, \sqrt{r_{11}^2+r_{21}^2}\right)$$
$$\psi = \mathrm{Atan2}\left(\frac{r_{21}}{\cos\theta}, \frac{r_{11}}{\cos\theta}\right)$$
$$\phi = \mathrm{Atan2}\left(\frac{r_{32}}{\cos\theta}, \frac{r_{33}}{\cos\theta}\right)$$

Angle axis $\theta\hat{\boldsymbol{w}}$:

$$\theta = \cos^{-1}\left(\frac{r_{11}+r_{22}+r_{33}-1}{2}\right)$$
$$\hat{\boldsymbol{w}} = \frac{1}{2\sin\theta}\begin{pmatrix} r_{32}-r_{23} \\ r_{13}-r_{31} \\ r_{21}-r_{12} \end{pmatrix}$$

Unit quaternions $(\epsilon_0\ \epsilon_1\ \epsilon_2\ \epsilon_3)^{\mathrm{T}}$:

$$\epsilon_0 = \frac{1}{2}\sqrt{1+r_{11}+r_{22}+r_{33}}$$
$$\epsilon_1 = \frac{r_{32}-r_{23}}{4\epsilon_0}$$
$$\epsilon_2 = \frac{r_{13}-r_{31}}{4\epsilon_0}$$
$$\epsilon_3 = \frac{r_{21}-r_{12}}{4\epsilon_0}$$

Fixed Angles

A vector of three angles can also denote the orientation of coordinate frame i relative to coordinate frame j when each angle represents a rotation about an axis of a fixed frame. Appropriately, such angles are referred to as fixed angles, and the order of the rotations must again accompany the angles to define the orientation. X-Y-Z fixed angles, denoted here as $(\psi,\theta,\phi)^{\mathrm{T}}$, are a common convention from among the, again, 12 different possible orders of rotations. Taking the moving frame i and

the fixed frame j to be initially coincident, ψ is the yaw rotation about the fixed $\hat{\boldsymbol{x}}_j$ axis, θ is the pitch rotation about the fixed $\hat{\boldsymbol{y}}_j$ axis, and ϕ is the roll rotation about the fixed $\hat{\boldsymbol{z}}_j$ axis. As can be seen by comparing the respective equivalent rotation matrices in Table 2.1 and the respective conversions in Table 2.2, a set of *X-Y-Z* fixed angles is exactly equivalent to the same set of *Z-Y-X* Euler angles ($\alpha = \phi$, $\beta = \theta$, and $\gamma = \psi$). This result holds in general such that three rotations about the three axes of a fixed frame define the same orientation as the same three rotations taken in the opposite order about the three axes of a moving frame. Likewise, all fixed-angle representations of orientations suffer from the singularity discussed for Euler angles. Also, the relationship between the time derivatives of fixed angles and the angular velocity vector is similar to the relationship for Euler angles.

Angle–Axis

A single angle θ in combination with a unit vector $\hat{\boldsymbol{w}}$ can also denote the orientation of coordinate frame i relative to coordinate frame j. In this case, frame i is rotated through the angle θ about an axis defined by the vector $\hat{\boldsymbol{w}} = (w_x\ w_y\ w_z)^{\mathrm{T}}$ relative to frame j. The vector $\hat{\boldsymbol{w}}$ is sometimes referred to as the equivalent axis of a finite rotation. The angle-axis representation, typically written as either $\theta\hat{\boldsymbol{w}}$ or $(\theta w_x\ \theta w_y\ \theta w_z)^{\mathrm{T}}$, is superabundant by one because it contains four parameters. The auxiliary relationship that resolves this is the unit magnitude of vector $\hat{\boldsymbol{w}}$. Even with this auxiliary relationship, the angle-axis representation is not unique because rotation through an angle of $-\theta$ about $-\hat{\boldsymbol{w}}$ is equivalent to a rotation through θ about $\hat{\boldsymbol{w}}$. Table 2.3 contains the conversions from the angle-axis representation to unit quaternions and vice versa. The conversions from these two representations to Euler angles or fixed angles can be easily found by using the conversions in Table 2.2 in conjunction with the equivalent rotation matrices in Table 2.1. Velocity relationships are more easily dealt with using the closely related quaternion representation.

Table 2.3 Conversions from angle-axis to unit quaternion representations of orientation and vice versa

Angle-axis $\theta\hat{\boldsymbol{w}}$ to unit quaternion $(\epsilon_0\ \epsilon_1\ \epsilon_2\ \epsilon_3)^{\mathrm{T}}$:
$\epsilon_0 = \cos\frac{\theta}{2}$
$\epsilon_1 = w_x \sin\frac{\theta}{2}$
$\epsilon_2 = w_y \sin\frac{\theta}{2}$
$\epsilon_3 = w_z \sin\frac{\theta}{2}$
Unit quaternion $(\epsilon_0\ \epsilon_1\ \epsilon_2\ \epsilon_3)^{\mathrm{T}}$ to angle-axis $\theta\hat{\boldsymbol{w}}$:
$\theta = 2\cos^{-1}\epsilon_0$
$w_x = \frac{\epsilon_1}{\sin\frac{\theta}{2}}$
$w_y = \frac{\epsilon_2}{\sin\frac{\theta}{2}}$
$w_z = \frac{\epsilon_3}{\sin\frac{\theta}{2}}$

Quaternions

The quaternion representation of orientation due to *Hamilton* [2.1], while largely superseded by the simpler vector representations of *Gibbs* [2.2] and *Graßmann* [2.3], is extremely useful for problems in robotics that result in representational singularities in the vector/matrix notation [2.4]. Quaternions do not suffer from singularities as Euler and fixed angles do.

A quaternion $\boldsymbol{\epsilon}$ is defined to have the form

$$\boldsymbol{\epsilon} = \epsilon_0 + \epsilon_1 i + \epsilon_2 j + \epsilon_3 k\ ,$$

where the components ϵ_0, ϵ_1, ϵ_2, and ϵ_3 are scalars, sometimes referred to as Euler parameters, and i, j, and k are operators. The operators are defined to satisfy the following combinatory rules

$$\begin{aligned} &ii = jj = kk = -1\ , \\ &ij = k\ , jk = i\ , ki = j\ , \\ &ji = -k\ , kj = -i\ , ik = -j\ . \end{aligned}$$

Two quaternions are added by adding the respective components separately, so the operators act as separators. The null element for addition is the quaternion $\mathbf{0} = 0 + 0i + 0j + 0k$, and quaternion sums are associative, commutative, and distributive. The null element for multiplication is $\mathbf{I} = 1 + 0i + 0j + 0k$, as can be seen using $\mathbf{I}\boldsymbol{\epsilon} = \boldsymbol{\epsilon}$ for any quaternion $\boldsymbol{\epsilon}$. Quaternion products are associative and distributive, but not commutative, and following the conventions of the operators and addition, have the form

$$\begin{aligned} \boldsymbol{ab} = {} & a_0b_0 - a_1b_1 - a_2b_2 - a_3b_3 \\ & + (a_0b_1 + a_1b_0 + a_2b_3 - a_3b_2)i \\ & + (a_0b_2 + a_2b_0 + a_3b_1 - a_1b_3)j \\ & + (a_0b_3 + a_3b_0 + a_1b_2 - a_2b_1)k\ . \end{aligned} \tag{2.7}$$

It is convenient to define the conjugate of a quaternion

$$\tilde{\boldsymbol{\epsilon}} = \epsilon_0 - \epsilon_1 i - \epsilon_2 j - \epsilon_3 k\ ,$$

so that

$$\boldsymbol{\epsilon}\tilde{\boldsymbol{\epsilon}} = \tilde{\boldsymbol{\epsilon}}\boldsymbol{\epsilon} = \epsilon_0^2 + \epsilon_1^2 + \epsilon_2^2 + \epsilon_3^2\ .$$

A unit quaternion can then be defined such that $\boldsymbol{\epsilon}\tilde{\boldsymbol{\epsilon}} = 1$. Often, ϵ_0 is referred to as the scalar part of the quaternion, and $(\epsilon_1\ \epsilon_2\ \epsilon_3)^{\mathrm{T}}$ is referred to as the vector part.

Unit quaternions are used to describe orientation, and the unit magnitude provides the auxiliary relationship to resolve the use of superabundant (four) coordinates. A vector is defined in quaternion notation as a quaternion with $\epsilon_0 = 0$. Thus, a vector $\boldsymbol{p} = (p_x\, p_y\, p_z)^{\mathrm{T}}$ can be expressed as a quaternion $\boldsymbol{p} = p_x i + p_y j + p_z k$. For any unit quaternion $\boldsymbol{\epsilon}$, the operation $\boldsymbol{\epsilon p \tilde{\epsilon}}$ performs a rotation of the vector $\boldsymbol{p}$ about the direction $(\epsilon_1\, \epsilon_2\, \epsilon_3)^{\mathrm{T}}$. This is clearly seen by expanding the operation $\boldsymbol{\epsilon p \tilde{\epsilon}}$ and comparing the results with the equivalent rotation matrix listed in Table 2.1. Also, as shown in Table 2.3, unit quaternions are closely related to the angle-axis representation. ϵ_0 corresponds (but is not equal) to the angle of rotation, while ϵ_1, ϵ_2, and ϵ_3 define the axis of rotation.

For velocity analysis, the time derivative of the quaternion can be related to the angular velocity vector as

$$\begin{pmatrix} \dot{\epsilon}_0 \\ \dot{\epsilon}_1 \\ \dot{\epsilon}_2 \\ \dot{\epsilon}_3 \end{pmatrix} = \frac{1}{2} \begin{pmatrix} -\epsilon_1 & -\epsilon_2 & -\epsilon_3 \\ \epsilon_0 & \epsilon_3 & -\epsilon_2 \\ -\epsilon_3 & \epsilon_0 & \epsilon_1 \\ \epsilon_2 & -\epsilon_1 & \epsilon_0 \end{pmatrix} \begin{pmatrix} \omega_x \\ \omega_y \\ \omega_z \end{pmatrix}, \tag{2.8}$$

where $(\omega_x, \omega_y, \omega_z)^{\mathrm{T}} = {}^j\boldsymbol{\omega}_i$ is given in fixed frame j. Defining $\boldsymbol{\epsilon}_{1:3} = (\epsilon_1, \epsilon_2, \epsilon_3)^{\mathrm{T}}$, it is straightforward to verify that

$$\dot{\epsilon}_0 = -\frac{1}{2}\, {}^j\boldsymbol{\omega}_i^{\mathrm{T}}\, \boldsymbol{\epsilon}_{1:3}$$
$$\dot{\boldsymbol{\epsilon}}_{1:3} = \frac{1}{2}\left(\epsilon_0\, {}^j\boldsymbol{\omega}_i - \boldsymbol{\epsilon}_{1:3} \times {}^j\boldsymbol{\omega}_i\right) .$$

In the case that the angular velocity is given in the moving coordinates, these rate equations take a similar form.

$$\dot{\epsilon}_0 = -\frac{1}{2}\, {}^i\boldsymbol{\omega}_i^{\mathrm{T}}\, \boldsymbol{\epsilon}_{1:3}$$
$$\dot{\boldsymbol{\epsilon}}_{1:3} = \frac{1}{2}\left(\epsilon_0\, {}^i\boldsymbol{\omega}_i + \boldsymbol{\epsilon}_{1:3} \times {}^i\boldsymbol{\omega}_i\right)$$

The corresponding rate matrix for this case is the same as (2.8) with the exception that the off-diagonal elements in the bottom 3×3 submatrix have opposite sign.

While a unit quaternion represents only the orientation of a body, quaternions may be dualized [2.5–7] to create an algebra that provides a description of the position and orientation of a body in space. Other combined representations are discussed in the following sections.

2.2.3 Homogeneous Transformations

The preceding sections have addressed representations of position and orientation separately. With homogeneous transformations, position vectors and rotation matrices are combined together in a compact notation. Any vector ${}^i\boldsymbol{r}$ expressed relative to the i coordinate frame can be expressed relative to the j coordinate frame if the position and orientation of the i frame are known relative to the j frame. Using the notation of Sect. 2.2.1, the position of the origin of coordinate frame i relative to coordinate frame j can be denoted by the vector ${}^j\boldsymbol{p}_i = ({}^jp_i^x\, {}^jp_i^y\, {}^jp_i^z)^{\mathrm{T}}$. Using the notation of Sect. 2.2.2, the orientation of frame i relative to frame j can be denoted by the rotation matrix ${}^j\mathbf{R}_i$. Thus,

$${}^j\boldsymbol{r} = {}^j\mathbf{R}_i\, {}^i\boldsymbol{r} + {}^j\boldsymbol{p}_i\, . \tag{2.9}$$

This equation can be written

$$\begin{pmatrix} {}^j\boldsymbol{r} \\ 1 \end{pmatrix} = \begin{pmatrix} {}^j\mathbf{R}_i & {}^j\boldsymbol{p}_i \\ \mathbf{0}^{\mathrm{T}} & 1 \end{pmatrix} \begin{pmatrix} {}^i\boldsymbol{r} \\ 1 \end{pmatrix}, \tag{2.10}$$

where

$${}^j\mathbf{T}_i = \begin{pmatrix} {}^j\mathbf{R}_i & {}^j\boldsymbol{p}_i \\ \mathbf{0}^{\mathrm{T}} & 1 \end{pmatrix} \tag{2.11}$$

is the 4×4 homogeneous transformation matrix and $({}^j\boldsymbol{r}\ 1)^{\mathrm{T}}$ and $({}^i\boldsymbol{r}\ 1)^{\mathrm{T}}$ are the homogeneous representations of the position vectors ${}^j\boldsymbol{r}$ and ${}^i\boldsymbol{r}$. The matrix ${}^j\mathbf{T}_i$ transforms vectors from coordinate frame i to coordinate frame j. Its inverse ${}^j\mathbf{T}_i^{-1}$ transforms vectors from coordinate frame j to coordinate frame i.

$${}^j\mathbf{T}_i^{-1} = {}^i\mathbf{T}_j = \begin{pmatrix} {}^j\mathbf{R}_i^{\mathrm{T}} & -{}^j\mathbf{R}_i^{\mathrm{T}}\, {}^j\boldsymbol{p}_i \\ \mathbf{0}^{\mathrm{T}} & 1 \end{pmatrix}. \tag{2.12}$$

Composition of 4×4 homogeneous transformation matrices is accomplished through simple matrix multiplication, just as in the case of 3×3 rotation matrices. Therefore, ${}^k\mathbf{T}_i = {}^k\mathbf{T}_j\, {}^j\mathbf{T}_i$. Since matrix multiplications do not commute, the order or sequence is important.

The homogeneous transformation of a simple rotation about an axis is sometimes denoted ***Rot*** such that a rotation of θ about an axis $\hat{\boldsymbol{z}}$ is

$$\mathbf{Rot}(\hat{\boldsymbol{z}}, \theta) = \begin{pmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{2.13}$$

Similarly, the homogeneous transformation of a simple translation along an axis is sometimes denoted ***Trans*** such that a translation of d along an axis $\hat{\boldsymbol{x}}$ is

$$\mathbf{Trans}(\hat{\boldsymbol{x}}, d) = \begin{pmatrix} 1 & 0 & 0 & d \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{2.14}$$

Homogeneous transformations are particularly attractive when compact notation is desired and/or when ease of programming is the most important consideration. This is not, however, a computationally efficient representation since it introduces a large number of additional multiplications by ones and zeros. Although homogeneous transformation matrices technically contain 16 elements, four are defined to be zero or one, and the remaining elements are composed of a rotation matrix and a position vector. Therefore, the only truly superabundant coordinates come from the rotation matrix component, so the relevant auxiliary relationships are those associated with the rotation matrix.

2.2.4 Screw Transformations

The transformation in (2.9) can be viewed as composed of a rotation between coordinate frames i and j and a separate displacement between those frames. To get from frame i to frame j, one could perform the rotation first, followed by the displacement, or vice versa. Alternatively, the spatial displacement between the frames can be expressed, except in the case of a pure translation, as a rotation about a unique line combined with a translation parallel to that line.

Chasles' Theorem

Chasles' theorem, in the form stated by *Chirikjian* and *Kyatkin* [2.8], has two parts. The first states that:

> *Any displacement of a body in space can be accomplished by means of a translation of a designated point from its initial to its final position, followed by a rotation of the whole body about that point to bring it into its final orientation.*

The second part states that:

> *Any displacement of a body in space can be accomplished by means of a rotation of the body about a unique line in space accompanied by a translation of the body parallel to that line.*

Such a line is called a screw axis, and it is this second result that is usually thought of as Chasles' theorem.

The first part of the theorem is almost axiomatic. A designated point in a body anywhere in Euclidean space can be displaced from a given initial position to a given final position. By further requiring that all points in the body traverse the same displacement, the body translates so that the designated point moves from its initial position to its final position. The body can then be rotated about that point into any given final orientation.

The second part of the theorem depends on this representation of a spatial displacement and requires a more complex argument. A preliminary theorem due to Euler allows greater specificity about the rotation of the body: *Any displacement of a body in which one point remains fixed is equivalent to a rotation of that body about a unique axis passing through that point.* Geometrically, embedding three points in the moving body and letting one be the fixed point about which rotation occurs, each of the other two will have initial and final positions. The right bisector planes of the lines joining the initial and final positions in each case necessarily contain the fixed point. Any line in the bisector plane can be the axis of a rotation that carries the corresponding point from its initial to its final position. Therefore, the unique line common to the two bisector planes is such that rotation about it will carry any point in the body from its initial to its final position. The rigidity condition requires that all planes in the body that contain that line rotate through the same angle.

For any rotation of a rigid body described by a rotation matrix ${}^{j}\mathbf{R}_{i}$, Euler's theorem states that there is a unique eigenvector $\hat{\boldsymbol{w}}$ such that

$$ {}^{j}\mathbf{R}_{i}\hat{\boldsymbol{w}} = \hat{\boldsymbol{w}} \,, \tag{2.15} $$

where $\hat{\boldsymbol{w}}$ is a unit vector parallel to the axis of rotation. This expression requires a unit eigenvalue of ${}^{j}\mathbf{R}_{i}$ corresponding to the eigenvector $\hat{\boldsymbol{w}}$. The remaining two eigenvalues are $\cos\theta \pm i\sin\theta$, where i is the complex operator and θ is the angle of rotation of the body about the axis.

Combining the first part of Chasles' theorem with Euler's theorem, a general spatial displacement can be expressed as a translation taking a point from its initial to its final position, followed by a unique rotation about a unique axis through that point that carries the body from its initial to its final orientation. Resolving the translation into components in the direction of and orthogonal to the rotation axis, every point in the body has the same displacement component in the direction of the axis because rotation about it does not affect that component. Projected onto a plane normal to the rotation axis, the kinematic geometry of the displacement is identical to that of planar motion. Just as there is a unique point in the plane about which a body could be rotated between two given positions, there is a unique point in the projection plane. If the rotation axis is moved to pass through that point, the spatial displacement can be accomplished by a rotation about that axis combined with a translation along it, as stated by the theorem.

The line about which rotation takes place is called the screw axis of the displacement. The ratio of the linear displacement d to the rotation θ is referred to as the

pitch h of the screw axis [2.4]. Thus,

$$d = h\theta \,. \tag{2.16}$$

The screw axis of a pure translation is not unique. Any line parallel to the translation can be regarded as the screw axis, and since the rotation θ is zero, the axis of a translation is said to have infinite pitch.

A screw axis is most conveniently represented in any coordinate frame by means of a unit vector $\hat{\boldsymbol{w}}$ parallel to it and the position vector $\boldsymbol{\rho}$ of any point lying on it. Additional specification of the pitch h and the angle of rotation θ completely defines the location of a second coordinate frame relative to the first frame. Thus, a total of eight coordinates define a screw transformation, which is superabundant by two. The unit magnitude of vector $\hat{\boldsymbol{w}}$ provides one auxiliary relationship, but in general, there is no second auxiliary relationship because the same screw axis is defined by all points lying on it, which is to say that the vector $\boldsymbol{\rho}$ contains one free coordinate.

Algebraically, a screw displacement is represented by

$$^j\boldsymbol{r} = {}^j\mathbf{R}_i(^i\boldsymbol{r} - \boldsymbol{\rho}) + d\hat{\boldsymbol{w}} + \boldsymbol{\rho} \,. \tag{2.17}$$

Comparing this expression to (2.9) yields

$$^j\boldsymbol{p}_i = d\hat{\boldsymbol{w}} + \left(\mathbf{1}_{3\times3} - {}^j\mathbf{R}_i\right)\boldsymbol{\rho} \,. \tag{2.18}$$

$\mathbf{1}_{3\times3}$ denotes the 3×3 identity matrix. An expression for d is easily obtained by taking the inner product of both sides of the equation with $\hat{\boldsymbol{w}}$.

$$d = \hat{\boldsymbol{w}}^{\mathrm{T}}{}^j\boldsymbol{p}_i \,. \tag{2.19}$$

The matrix $\mathbf{1}_{3\times3} - {}^j\mathbf{R}_i$ is singular, so (2.18) cannot be solved to give a unique value of $\boldsymbol{\rho}$, but since $\boldsymbol{\rho}$ can represent any point on the screw axis, this would not be appropriate. One component of $\boldsymbol{\rho}$ can be arbitrarily chosen, and any two of the component equations can then be solved to find the two other components of $\boldsymbol{\rho}$. All other points on the screw axis are then given by $\boldsymbol{\rho} + k\hat{\boldsymbol{w}}$, where k can take any value.

Table 2.4 contains the conversions between screw transformations and homogeneous transformations. Note that the equivalent rotation matrix for a screw transformation has the same form as the equivalent rotation matrix for an angle-axis representation of orientation in Table 2.1. Also, the auxiliary relationship that the vector $\boldsymbol{\rho}$ be orthogonal to the screw axis ($\hat{\boldsymbol{w}}^{\mathrm{T}}\boldsymbol{\rho} = 0$) is used in Table 2.4 to provide a unique conversion to the screw transformation. The inverse result, that of finding the rotation matrix ${}^j\mathbf{R}_i$ and the translation ${}^j\boldsymbol{p}_i$ corresponding to a given screw displacement, is found from Rodrigues' equation.

Rodrigues' Equation

Given a screw axis, the angular displacement of a body about it, and the translation of the body along it, the displacement of an arbitrary point in that body can be found. Viewing a matrix transformation as describing the displacement of the body, this is equivalent to finding the matrix transformation equivalent to a given screw displacement.

Referring to Fig. 2.1, the position vectors of a point before and after a screw displacement can be geometrically related as

$$\begin{aligned} ^j\boldsymbol{r} = {}^i\boldsymbol{r} + d\hat{\boldsymbol{w}} + \sin\theta\,\hat{\boldsymbol{w}} \times (^i\boldsymbol{r} - \boldsymbol{\rho}) \\ - (1 - \cos\theta)(^i\boldsymbol{r} - \boldsymbol{\rho}) - (^i\boldsymbol{r} - \boldsymbol{\rho}) \cdot \hat{\boldsymbol{w}}\,\hat{\boldsymbol{w}} \,, \end{aligned} \tag{2.20}$$

where ${}^i\boldsymbol{r}$ and ${}^j\boldsymbol{r}$ denote the initial and final positions of the point, $\hat{\boldsymbol{w}}$ and $\boldsymbol{\rho}$ specify the screw axis, and θ and d give the displacement about it. This result is usually referred to as Rodrigues' equation [2.9], which can be written as a matrix transformation [2.10],

$$^j\boldsymbol{r} = {}^j\mathbf{R}_i{}^i\boldsymbol{r} + {}^j\boldsymbol{p}_i \,, \tag{2.21}$$

since, when expanded, it gives three linear equations for the components of ${}^j\boldsymbol{r}$ in terms of those of ${}^i\boldsymbol{r}$.

$$^j\mathbf{R}_i = \begin{pmatrix} w_x^2 v_\theta + c_\theta & w_x w_y v_\theta - w_z s_\theta & w_x w_z v_\theta + w_y s_\theta \\ w_x w_y v_\theta + w_z s_\theta & w_y^2 v_\theta + c_\theta & w_y w_z v_\theta - w_x s_\theta \\ w_x w_z v_\theta - w_y s_\theta & w_y w_z v_\theta + w_x s_\theta & w_z^2 v_\theta + c_\theta \end{pmatrix}$$

$$^j\boldsymbol{p}_i = \left(\mathbf{1}_{3\times3} - {}^j\mathbf{R}_i\right)\boldsymbol{\rho} + h\theta\hat{\mathbf{w}} \,,$$

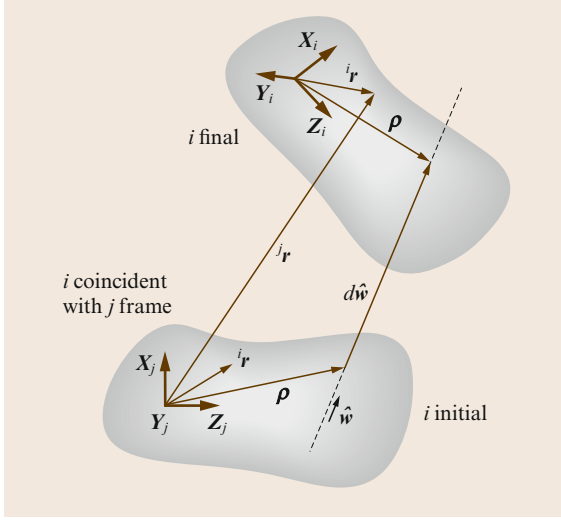


Fig. 2.1 Initial and final positions of an arbitrary point in a body undergoing a screw displacement; ${}^i\boldsymbol{r}$ is the position of the point relative to the moving frame, which is coincident with the fixed frame j in its initial position; ${}^j\boldsymbol{r}$ is the position of the point relative to the fixed frame after the screw displacement of the moving body

Table 2.4 Conversions from a screw transformation to a homogeneous transformation and vice versa, with abbreviations $c_\theta := \cos\theta$, $s_\theta := \sin\theta$, and $v_\theta := 1 - \cos\theta$

Screw transformation to homogeneous transformation	${}^j\mathbf{R}_i = \begin{pmatrix} w_x^2 v_\theta + c_\theta & w_x w_y v_\theta - w_z s_\theta & w_x w_z v_\theta + w_y s_\theta \\ w_x w_y v_\theta + w_z s_\theta & w_y^2 v_\theta + c_\theta & w_y w_z v_\theta - w_x s_\theta \\ w_x w_z v_\theta - w_y s_\theta & w_y w_z v_\theta + w_x s_\theta & w_z^2 v_\theta + c_\theta \end{pmatrix}$ ${}^j\mathbf{p}_i = (\mathbf{1}_{3\times3} - {}^j\mathbf{R}_i)\boldsymbol{\rho} + h\theta\,\hat{\boldsymbol{w}}$
Homogeneous transformation to screw transformation	$\boldsymbol{l} = \begin{pmatrix} r_{32} - r_{23} \\ r_{13} - r_{31} \\ r_{21} - r_{12} \end{pmatrix}^{\mathrm{T}}$ $\theta = \text{sign}\left(\boldsymbol{l}^{\mathrm{T}} {}^j\boldsymbol{p}_i\right) \left\lvert \cos^{-1}\left(\frac{r_{11} + r_{22} + r_{33} - 1}{2}\right)\right\rvert$ $h = \frac{\boldsymbol{l}^{\mathrm{T}} {}^j\boldsymbol{p}_i}{2\theta \sin\theta}$ $\boldsymbol{\rho} = \frac{(\mathbf{1}_{3\times3} - {}^j\mathbf{R}_i^{\mathrm{T}}) {}^j\boldsymbol{p}_i}{2(1-\cos\theta)}$ $\hat{\boldsymbol{w}} = \frac{\boldsymbol{l}}{2\sin\theta}$

where the abbreviations are $c_\theta := \cos\theta$, $s_\theta := \sin\theta$, and $v_\theta = 1 - \cos\theta$. The rotation matrix ${}^j\mathbf{R}_i$ expressed in this form is also called the screw matrix, and these equations give the elements of ${}^j\mathbf{R}_i$ and ${}^j\boldsymbol{p}_i$ in terms of the screw parameters.

An exception arises in the case of a pure translation, for which $\theta = 0$ and Rodrigues' equation becomes

$$ {}^j\boldsymbol{r} = {}^i\boldsymbol{r} + d\,\hat{\boldsymbol{w}} \,. \tag{2.22}$$

Substituting for this case, ${}^j\mathbf{R}_i = \mathbf{1}_{3\times3}$ and ${}^j\boldsymbol{p}_i = d\,\hat{\boldsymbol{w}}$.

Additional information on screw theory can be found in [2.11–15].

2.2.5 Matrix Exponential Parameterization

The position and orientation of a body can also be expressed in a unified fashion with an exponential mapping. This approach is introduced first with its application to pure rotation and expanded to rigid-body motion. More details on the approach can be found in [2.16] and [2.17].

Exponential Coordinates for Rotation

The set of all orthogonal matrices with determinant 1, which is the set of all rotation matrices $\mathbf{R}$, is a *group* under the operation of matrix multiplication denoted as $SO(3) \subset \mathbb{R}^{3\times3}$ [2.18]. This stands for *special orthogonal* wherein *special* alludes to the det $\mathbf{R}$ being $+1$ instead of ±1. The set of rotation matrices satisfies the four axioms of a group:

- *Closure*: $\mathbf{R}_1\mathbf{R}_2 \in SO(3) \quad \forall \mathbf{R}_1, \mathbf{R}_2 \in SO(3)$
- *Identity*: $\mathbf{1}_{3\times3}\mathbf{R} = \mathbf{R}\mathbf{1}_{3\times3} = \mathbf{R} \quad \forall \mathbf{R} \in SO(3)$
- *Inverse*: $\mathbf{R}^{\mathrm{T}} \in SO(3)$ is the unique inverse of $\mathbf{R}$ $\forall \mathbf{R} \in SO(3)$
- *Associativity*: $(\mathbf{R}_1\mathbf{R}_2)\mathbf{R}_3 = \mathbf{R}_1(\mathbf{R}_2\mathbf{R}_3) \quad \forall \mathbf{R}_1, \mathbf{R}_2, \mathbf{R}_3 \in SO(3)$.

In the angle-axis representation presented in Sect. 2.2.2, orientation is expressed as an angle θ of rotation about an axis defined by the unit vector $\hat{\boldsymbol{w}}$. The equivalent rotation matrix found in Table 2.1 can be expressed as the exponential map

$$\begin{aligned} \mathbf{R} &= \mathrm{e}^{\mathbf{S}(\hat{\boldsymbol{w}})\theta} \\ &= \mathbf{1}_{3\times3} + \theta\mathbf{S}(\hat{\boldsymbol{w}}) + \frac{\theta^2}{2!}\mathbf{S}(\hat{\boldsymbol{w}})^2 \\ &\quad + \frac{\theta^3}{3!}\mathbf{S}(\hat{\boldsymbol{w}})^3 + \ldots \,, \end{aligned} \tag{2.23}$$

where $\mathbf{S}(\hat{\boldsymbol{w}})$ is the unit skew-symmetric matrix

$$\mathbf{S}(\hat{\boldsymbol{w}}) = \begin{pmatrix} 0 & -w_z & w_y \\ w_z & 0 & -w_x \\ -w_y & w_x & 0 \end{pmatrix} . \tag{2.24}$$

Thus, the exponential map transforms a skew-symmetric matrix $\mathbf{S}(\hat{\boldsymbol{w}})$ that corresponds to an axis of rotation $\hat{\boldsymbol{w}}$ into an orthogonal matrix $\mathbf{R}$ that corresponds to a rotation about the axis $\hat{\boldsymbol{w}}$ through an angle of θ. It can be shown that the closed-form expression for $\mathrm{e}^{\mathbf{S}(\hat{\boldsymbol{w}})\theta}$, which can be efficiently computed, is

$$\mathrm{e}^{\mathbf{S}(\hat{\boldsymbol{w}})\theta} = \mathbf{1}_{3\times3} + \mathbf{S}(\hat{\boldsymbol{w}})\sin\theta + \mathbf{S}(\hat{\boldsymbol{w}})^2(1 - \cos\theta) \,. \tag{2.25}$$

The components of $(\theta w_x\ \theta w_y\ \theta w_z)^{\mathrm{T}}$, which are related to the elements of the rotation matrix $\mathbf{R}$ in Table 2.2, are referred to as the *exponential coordinates* for $\mathbf{R}$.

Exponential Coordinates for Rigid-Body Motion

As indicated in Sect. 2.2.3, the position and orientation of a body can be expressed by the combination of a position vector $\boldsymbol{p} \in \mathbb{R}^3$ and a rotation matrix $\mathbf{R} \in SO(3)$. The product space of $\mathbb{R}^3$ with $SO(3)$ is the group known as $SE(3)$, which stands for *special Euclidean*.

$$SE(3) = \{(\boldsymbol{p}, \mathbf{R}) : \boldsymbol{p} \in \mathbb{R}^3, \mathbf{R} \in SO(3)\} = \mathbb{R}^3 \times SO(3) .$$

The set of homogeneous transformations satisfies the four axioms of a group:

- *Closure*: $\mathbf{T}_1\mathbf{T}_2 \in SE(3) \quad \forall \mathbf{T}_1, \mathbf{T}_2 \in SE(3)$
- *Identity*: $\mathbf{1}_{4\times4}\mathbf{T} = \mathbf{T}\mathbf{1}_{4\times4} = \mathbf{T} \quad \forall \mathbf{T} \in SE(3)$
- *Inverse*: the unique inverse of $\mathbf{T} \quad \forall \mathbf{T} \in SE(3)$ is given in (2.12)
- *Associativity*: $(\mathbf{T}_1\mathbf{T}_2)\mathbf{T}_3 = \mathbf{T}_1(\mathbf{T}_2\mathbf{T}_3) \quad \forall \mathbf{T}_1, \mathbf{T}_2, \mathbf{T}_3 \in SE(3)$.

In the screw transformation representation in Sect. 2.2.4, position and orientation are expressed by the angle θ of rotation about a screw axis defined by the unit vector $\hat{\boldsymbol{w}}$, the point $\boldsymbol{\rho}$ on the axis such that $\hat{\boldsymbol{w}}^{\mathrm{T}}\boldsymbol{\rho} = 0$, and the pitch h of the screw axis. The equivalent homogeneous transformation found in Table 2.4 can be expressed as the exponential map

$$\mathbf{T} = \mathrm{e}^{\hat{\boldsymbol{\xi}}\theta} = \mathbf{1}_{4\times4} + \hat{\boldsymbol{\xi}}\theta + \frac{(\hat{\boldsymbol{\xi}}\theta)^2}{2!} + \frac{(\hat{\boldsymbol{\xi}}\theta)^3}{3!} + \dots , \tag{2.26}$$

where

$$\hat{\boldsymbol{\xi}} = \begin{pmatrix} S(\hat{\boldsymbol{w}}) & \boldsymbol{v} \\ \mathbf{0}^{\mathrm{T}} & 0 \end{pmatrix} \tag{2.27}$$

is the generalization of the unit skew-symmetric matrix $\mathbf{S}(\hat{\boldsymbol{w}})$ known as a *twist*. The twist coordinates of $\hat{\boldsymbol{\xi}}$ are given by $\boldsymbol{\xi} := (\hat{\boldsymbol{w}}^{\mathrm{T}}\boldsymbol{v}^{\mathrm{T}})^{\mathrm{T}}$. It can be shown that the closed-form expression for $\mathrm{e}^{\hat{\boldsymbol{\xi}}\theta}$ is

$$\mathrm{e}^{\hat{\boldsymbol{\xi}}\theta} = \begin{pmatrix} \mathrm{e}^{\mathbf{S}(\hat{\boldsymbol{w}})\theta} & (\mathbf{1}_{3\times3} - \mathrm{e}^{\mathbf{S}(\hat{\boldsymbol{w}})\theta})(\hat{\boldsymbol{w}} \times \boldsymbol{v}) + \hat{\boldsymbol{w}}^{\mathrm{T}}\boldsymbol{v}\theta\hat{\boldsymbol{w}} \\ \mathbf{0}^{\mathrm{T}} & 1 \end{pmatrix} . \tag{2.28}$$

Comparison of this result with the conversion between homogeneous and screw transformations in Table 2.4 yields

$$\boldsymbol{v} = \boldsymbol{\rho} \times \hat{\boldsymbol{w}} \tag{2.29}$$

and

$$h = \hat{\boldsymbol{w}}^{\mathrm{T}}\boldsymbol{v} . \tag{2.30}$$

Thus, the exponential map for a twist transforms the initial pose of a body into its final pose. It gives the relative rigid-body motion. The vector $\boldsymbol{\xi}\theta$ contains the *exponential coordinates* for the rigid-body transformation.

As for screw transformations, the case of pure translation is unique. In this case, $\hat{\boldsymbol{w}} = \mathbf{0}$, so

$$\mathrm{e}^{\hat{\boldsymbol{\xi}}\theta} = \begin{pmatrix} \mathbf{1}_{3\times3} & \theta\boldsymbol{v} \\ \mathbf{0}^{\mathrm{T}} & 1 \end{pmatrix} . \tag{2.31}$$

2.2.6 Plücker Coordinates

A minimum of four coordinates are needed to define a line in space. The Plücker coordinates of a line form a six-dimensional (6-D) vector, so they are superabundant by two. They can be viewed as a pair of three-dimensional (3-D) vectors; one is parallel to the line, and the other is the *moment* of that vector about the origin. Thus, if $\boldsymbol{u}$ is any vector parallel to the line and $\boldsymbol{\rho}$ is the position of any point on the line relative to the origin, the Plücker coordinates (L, M, N, P, Q, and R) are given by

$$(L, M, N) = \boldsymbol{u}^{\mathrm{T}}; \; (P, Q, R) = (\boldsymbol{\rho} \times \boldsymbol{u})^{\mathrm{T}} . \tag{2.32}$$

For simply defining a line, the magnitude of $\boldsymbol{u}$ is not unique, nor is the component of $\boldsymbol{\rho}$ parallel to $\boldsymbol{u}$. Two auxiliary relationships are imposed to reduce the set to just four independent coordinates. One is that the scalar product of the two three-dimensional vectors is identically zero.

$$LP + MQ + NR \equiv 0 . \tag{2.33}$$

The other is the invariance of the line designated when the coordinates are all multiplied by the same scaling factor.

$$(L, M, N, P, Q, R) \equiv (kL, kM, kN, kP, kQ, kR) . \tag{2.34}$$

This relationship may take the form of constraining $\boldsymbol{u}$ to have unit magnitude so that L, M, and N are the direction cosines.

In this handbook, it is often useful to express velocities in Plücker coordinates, wherein unlike the definition of lines, the magnitudes of the two three-dimensional vectors are not arbitrary. This leads to the motor notation of *von Mises* [2.9, 19] and *Everett* [2.20]. For instantaneously coincident coordinate frames, one fixed and the other embedded in the moving body, $\boldsymbol{\omega}$ is the angular velocity of the body and $\boldsymbol{v}_O$ is the velocity of the origin O of the body-fixed frame when both are expressed relative to the fixed frame. This provides a Plücker coordinate system for the spatial velocity $\mathbf{v}$ of the body. The Plücker coordinates of $\mathbf{v}$

are simply the Cartesian coordinates of $\boldsymbol{\omega}$ and $\boldsymbol{v}_O$,

$$\mathbf{v} = \begin{pmatrix} \boldsymbol{\omega} \\ \boldsymbol{v}_O \end{pmatrix} . \tag{2.35}$$

The transformation from Plücker coordinate system i to Plücker coordinate system j for spatial velocities is achieved with the spatial transform ${}^{j}\mathbf{X}_i$. If $\mathbf{v}_i$ and $\mathbf{v}_j$ denote the spatial velocities of a body relative to the i and j frames, respectively, and ${}^{j}\boldsymbol{p}_i$ and ${}^{j}\mathbf{R}_i$ denote the position and orientation of frame i relative to frame j,

$$\mathbf{v}_j = {}^{j}\mathbf{X}_i \mathbf{v}_i \, , \tag{2.36}$$

where

$${}^{j}\mathbf{X}_i = \begin{pmatrix} {}^{j}\mathbf{R}_i & \mathbf{0}_{3\times3} \\ \mathbf{S}({}^{j}\boldsymbol{p}_i)\,{}^{j}\mathbf{R}_i & {}^{j}\mathbf{R}_i \end{pmatrix} , \tag{2.37}$$

such that

$${}^{j}\mathbf{X}_i{}^{-1} = {}^{i}\mathbf{X}_j = \begin{pmatrix} {}^{i}\mathbf{R}_j & \mathbf{0}_{3\times3} \\ -{}^{i}\mathbf{R}_j\mathbf{S}({}^{j}\boldsymbol{p}_i) & {}^{i}\mathbf{R}_j \end{pmatrix} , \tag{2.38}$$

and

$${}^{k}\mathbf{X}_i = {}^{k}\mathbf{X}_j {}^{j}\mathbf{X}_i \, , \tag{2.39}$$

and $\mathbf{S}({}^{j}\boldsymbol{p}_i)$ is the skew-symmetric matrix

$$\begin{pmatrix} 0 & -{}^{j}p_i^z & {}^{j}p_i^y \\ {}^{j}p_i^z & 0 & -{}^{j}p_i^x \\ -{}^{j}p_i^y & {}^{j}p_i^x & 0 \end{pmatrix} . \tag{2.40}$$

Spatial vector notation, which includes the spatial velocities and transforms briefly mentioned here, is treated in greater depth in Sect. 3.2. Specifically, Table 3.1 gives a computationally efficient algorithm for applying a spatial transform.

2.3 Joint Kinematics

Unless explicitly stated otherwise, the kinematic description of robotic mechanisms typically employs a number of idealizations. The links that compose the robotic mechanism are assumed to be perfectly rigid bodies having surfaces that are geometrically perfect in both position and shape. Accordingly, these rigid bodies are connected together at joints where their idealized surfaces are in ideal contact without any clearance between them. The respective geometries of these surfaces in contact determine the freedom of motion between the two links, or the *joint kinematics*.

A kinematic joint is a connection between two bodies that constrains their relative motion. Two bodies that are in contact with one another create a simple kinematic joint. The surfaces of the two bodies that are in contact are able to move over one another, thereby permitting relative motion of the two bodies. Simple kinematic joints are classified as lower pair joints if contact occurs over surfaces [2.21] and as higher pair joints if contact occurs only at points or along lines.

A joint model describes the motion of a frame fixed in one body of a joint relative to a frame fixed in the other body. The motion is expressed as a function of the joint's motion variables, and other elements of a joint model include the rotation matrix, position vector, free modes, and constrained modes. The free modes of a joint define the directions in which motion is allowed. They are represented by the $6 \times n_i$ matrix $\boldsymbol{\Phi}_i$ whose columns are the Plücker coordinates of the allowable motion. This matrix relates the spatial velocity vector across the joint $\boldsymbol{v}_{\mathrm{rel,i}}$ to the joint velocity vector $\dot{\boldsymbol{q}}_i$,

$$\boldsymbol{v}_{\mathrm{rel,i}} = \boldsymbol{\Phi}_i \dot{\boldsymbol{q}}_i \, . \tag{2.41}$$

In contrast, the constrained modes of a joint define the directions in which motion is not allowed. They are represented by the $6 \times (6 - n_i)$ matrix $\boldsymbol{\Phi}_i^{\mathrm{c}}$ that is complementary to $\boldsymbol{\Phi}_i$. Tables 2.5 and 2.6 contain the formulas of the joint models for all of the joints described in this section. They are used extensively for the dynamic analysis presented in Chap. 3. Additional information on joints can be found in Chap. 4.

2.3.1 Lower Pair Joints

Lower pair joints are mechanically attractive since wear is spread over the whole surface and lubricant is trapped in the small clearance space (in nonidealized systems) between the surfaces, resulting in relatively good lubrication. As can be proved [2.23] from the requirement for surface contact, there are only six possible forms of lower pair joints: revolute, prismatic, helical, cylindrical, spherical, and planar joints.

Revolute

The most general form of a revolute joint, often abbreviated as *R* and sometimes referred to colloquially as a hinge or pin joint, is a lower pair composed of two

Table 2.5 Joint model formulas for one-degree-of-freedom lower pair joints, with abbreviations $c_{\theta_i} := \cos\theta_i$ and $s_{\theta_i} := \sin\theta_i$ (adapted in part from Table 4.1 in [2.22])

Joint type	Joint rotation matrix ${}^j\mathbf{R}_i$	Position vector ${}^j\boldsymbol{p}_i$	Free modes $\boldsymbol{\Phi}_i$	Constrained modes $\boldsymbol{\Phi}_i^c$	Pose state vars.	$\dot{\boldsymbol{q}}_i$
Revolute R	$\begin{pmatrix} c_{\theta_i} & -s_{\theta_i} & 0 \\ s_{\theta_i} & c_{\theta_i} & 0 \\ 0 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$	θ_i	$\dot{\theta}_i$
Prismatic P	$\mathbf{1}_{3\times3}$	$\begin{pmatrix} 0 \\ 0 \\ d_i \end{pmatrix}$	$\begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$	d_i	$\dot{d}_i$
Helical H (pitch h)	$\begin{pmatrix} c_{\theta_i} & -s_{\theta_i} & 0 \\ s_{\theta_i} & c_{\theta_i} & 0 \\ 0 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} 0 \\ 0 \\ h\theta_i \end{pmatrix}$	$\begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ h \end{pmatrix}$	$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -h \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$	θ_i	$\dot{\theta}_i$

congruent surfaces of revolution. The surfaces are the same except one of them is an external surface, convex in any plane normal to the axis of revolution, and one is an internal surface, concave in any plane normal to the axis. The surfaces may not be solely in the form of right circular cylinders, since surfaces of that form do not provide any constraint on axial sliding. A revolute joint permits only rotation of one of the bodies joined relative to the other. The position of one body relative to the other may be expressed as the angle between two lines normal to the joint axis, one fixed in each body. Thus, the joint has one *degree of freedom* (DOF). When the $\hat{\boldsymbol{z}}$ axis of coordinate frame i is aligned with a revolute joint axis, the formulas in Table 2.5 define the revolute joint model.

Prismatic

The most general form of a prismatic joint, often abbreviated as *P* and sometimes referred to colloquially as a sliding joint, is a lower pair formed from two congruent general cylindrical surfaces. These may not be right circular cylindrical surfaces. A general cylindrical surface is obtained by extruding any curve in a constant direction. Again, one surface is internal and the other is an external surface. A prismatic joint permits only sliding of one of the members joined relative to the other along the direction of extrusion. The position of one body relative to the other is determined by the distance between two points on a line parallel to the direction of sliding, with one point fixed in each body. Thus, this joint also has one degree of freedom. When the $\hat{\boldsymbol{z}}$ axis of coordinate frame i is aligned with a prismatic joint axis, the formulas in Table 2.5 define the prismatic joint model.

Helical

The most general form of a helical joint, often abbreviated as *H* and sometimes referred to colloquially as a screw joint, is a lower pair formed from two helicoidal surfaces formed by extruding any curve along a helical path. The simple example is a bolt and nut wherein the basic generating curve is a pair of straight lines. The angle θ of rotation about the axis of the helical joint is directly related to the distance d of displacement of one body relative to the other along that axis by the expression $d = h\theta$, where the constant h is called the pitch of the helical joint. When the $\hat{\boldsymbol{z}}$ axis of coordinate frame i is aligned with a helical joint axis, the formulas in Table 2.5 define the helical joint model.

Cylindrical

A cylindrical joint, often abbreviated as C, is a lower pair formed by contact of two congruent right circular cylinders, one an internal surface and the other an external surface. It permits both rotation about the cylinder axis and sliding parallel to it. Therefore, it is a joint with two degrees of freedom. Lower pair joints with more than one degree of freedom are easily replaced by kinematically equivalent compound joints (Sect. 2.3.3) that are serial chains of one-degree-of-freedom lower pairs. In the present case, the cylindrical joint can be replaced by a revolute in series with a pris-

Table 2.6 Joint model formulas for higher-degree-of-freedom lower pair joints, universal joint, rolling contact joint, and 6-DOF joint, with abbreviations $c_{\theta_i} := \cos\theta_i$ and $s_{\theta_i} := \sin\theta_i$ (adapted in part from Table 4.1 in [2.22]) *The Euler angles α_i, β_i, and γ_i could be used in place of the unit quaternion $\boldsymbol{\epsilon}_i$ to represent orientation

Joint type	Joint rotation matrix ${}^jR_i$	Position vector ${}^jp_i$	Free modes Φ_i	Constrained modes Φ_i^c	Pose Variables	Velocity Variables $\dot{q}_i$
Cylindrical C	$\begin{pmatrix} c_{\theta_i} & -s_{\theta_i} & 0 \\ s_{\theta_i} & c_{\theta_i} & 0 \\ 0 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} 0 \\ 0 \\ d_i \end{pmatrix}$	$\begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}$	θ_i d_i	$\begin{pmatrix} \dot{\theta}_i \\ \dot{d}_i \end{pmatrix}$
Spherical* S	$\begin{pmatrix} \text{Table 2.1} \end{pmatrix}$	$\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$	$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$	$\boldsymbol{\epsilon}_i$	$\boldsymbol{\omega}_{i\,\mathrm{rel}}$
Planar	$\begin{pmatrix} c_{\theta_i} & -s_{\theta_i} & 0 \\ s_{\theta_i} & c_{\theta_i} & 0 \\ 0 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} c_{\theta_i} d_{xi} - s_{\theta_i} d_{yi} \\ s_{\theta_i} d_{xi} + c_{\theta_i} d_{yi} \\ 0 \end{pmatrix}$	$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$	θ_i d_{xi} d_{yi}	$\begin{pmatrix} \dot{\theta}_i \\ \dot{d}_{xi} \\ \dot{d}_{yi} \end{pmatrix}$
Flat planar rolling contact (fixed radius r)	$\begin{pmatrix} c_{\theta_i} & -s_{\theta_i} & 0 \\ s_{\theta_i} & c_{\theta_i} & 0 \\ 0 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} r\theta_i c_{\theta_i} - r s_{\theta_i} \\ -r\theta_i s_{\theta_i} - r c_{\theta_i} \\ 0 \end{pmatrix}$	$\begin{pmatrix} 0 \\ 0 \\ 1 \\ r \\ 0 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & -r & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$	θ_i	$\dot{\theta}_i$
Universal U	$\begin{pmatrix} c_{\alpha_i} c_{\beta_i} & -s_{\alpha_i} & c_{\alpha_i} s_{\beta_i} \\ s_{\alpha_i} c_{\beta_i} & c_{\alpha_i} & s_{\alpha_i} s_{\beta_i} \\ -s_{\beta_i} & 0 & c_{\beta_i} \end{pmatrix}$	$\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$	$\begin{pmatrix} -s_{\beta_i} & 0 \\ 0 & 1 \\ c_{\beta_i} & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{pmatrix}$	$\begin{pmatrix} c_{\beta_i} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ s_{\beta_i} & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$	α_i β_i	$\begin{pmatrix} \dot{\alpha}_i \\ \dot{\beta}_i \end{pmatrix}$
6-DOF*	$\begin{pmatrix} \text{see} \\ \text{Table 2.1} \end{pmatrix}$	${}^0p_i$	$\mathbf{1}_{6\times 6}$		$\boldsymbol{\varepsilon}_i$ ${}^0p_i$	$\begin{pmatrix} \boldsymbol{\omega}_i \\ v_i \end{pmatrix}$

matic joint whose direction of sliding is parallel to the revolute axis. While simpler to implement using the geometric representation discussed in Sect. 2.4, this approach has disadvantages for dynamic simulation. Modeling a single cylindrical joint as a combination of a prismatic and a revolute joint requires the addition of a virtual link between the two with zero mass and zero length. The massless link can create computational problems. When the $\hat{z}$ axis of coordinate frame i is aligned with a cylindrical joint axis, the formulas in Table 2.6 define the cylindrical joint model.

Spherical

A spherical joint, often abbreviated as S, is a lower pair formed by contact of two congruent spherical surfaces. Once again, one is an internal surface, and the other is an external surface. A spherical joint permits rotation about any line through the center of the sphere. Thus, it permits independent rotation about axes in up to three different directions and has three degrees of freedom. A spherical joint is easily replaced by a kinematically equivalent compound joint consisting of three revolutes that have axes that all intersect in a single point – the center of the spherical joint. The revolute joint axes

do not need to be successively orthogonal, but often they are implemented that way. The arrangement is, in general, kinematically equivalent to a spherical joint, but it does exhibit a singularity when the revolute joint axes become coplanar. This is as compared to the native spherical joint that never has such a singularity. Likewise, if a spherical joint is modeled in simulation as three revolutes, computational difficulties again can arise from the necessary inclusion of massless virtual links having zero length. The joint model formulas of a spherical joint are given in Table 2.6.

Planar

A planar joint is formed by planar contacting surfaces. Like the spherical joint, it is a lower pair joint with three degrees of freedom. A kinematically equivalent compound joint consisting of a serial chain of three revolutes with parallel axes can replace a planar joint. As was the case with the spherical joint, the compound joint exhibits a singularity when the revolute axes become coplanar. When the $\hat{z}$ axis of coordinate frame i is aligned with the normal to the plane of contact, the formulas in Table 2.6 define the planar joint model.

2.3.2 Higher Pair Joints

Some higher pair joints also have attractive properties, particularly rolling pairs in which one body rolls without slipping over the surface of the other. This is mechanically attractive since the absence of sliding means the absence of abrasive wear. However, since ideal contact occurs at a point, or along a line, application of a load across the joint may lead to very high local stresses resulting in other forms of material failure and, hence, wear. Higher pair joints can be used to create kinematic joints with special geometric properties, as in the case of a gear pair or a cam and follower pair.

Rolling Contact

Rolling contact actually encompasses several different geometries. Rolling contact in planar motion permits one degree of freedom of relative motion as in the case of a roller bearing, for example. Planar rolling contact can take place along a line, thereby spreading the load and wear somewhat. Three-dimensional rolling contact allows rotation about any axis through the point of contact that is, in principle, unique. Hence, a three-dimensional rolling contact pair permits relative motion with three degrees of freedom. When the $\hat{z}$ axis of coordinate frame i is aligned with the axis of rotation and passes through the center of the roller of fixed radius r, the formulas in Table 2.6 define the planar rolling contact joint model for a roller on a flat surface.

Regardless of whether the joint is planar or three-dimensional, the *no-slip* condition associated with a rolling contact joint requires that the instantaneous relative velocity between the points on the two bodies in contact be zero. If P is the point of rolling contact between bodies i and j,

$$v_{P_i/P_j} = 0\,. \tag{2.42}$$

Likewise, relative acceleration is in the direction of the common normal to the two surfaces at the point of contact. Because the constraint associated with the joint is expressed in terms of velocity and cannot be expressed in terms of position alone, it is nonholonomic, as discussed in Sect. 2.3.6. A more detailed discussion of the kinematic constraints for rolling contact is found in Chap. 24.

2.3.3 Compound Joints

Compound kinematic joints are connections between two bodies formed by chains of other members and simple kinematic joints. A compound joint may constrain the relative motion of the two bodies joined in the same way as a simple joint. In such a case, the two joints are said to be kinematically equivalent.

Universal

A universal joint, often abbreviated as *U* and referred to as a Cardan or Hooke joint, is a compound joint with two degrees of freedom. It consists of a serial chain of two revolutes whose axes intersect orthogonally. The joint model for a universal joint, in which, from Euler angle notation, α_i is the first rotation about the Z-axis and then β_i is the rotation about the Y-axis, is given in Table 2.6. This is a joint for which the matrices $\mathbf{\Phi}_i$ and $\mathbf{\Phi}_i^c$ are not constant, so in general, $\dot{\mathbf{\Phi}}_i \neq 0$ and $\dot{\mathbf{\Phi}}_i^c \neq 0$. As seen in Table 2.6, the orientation of the first joint axis (expressed in the outboard coordinate frame) varies with β_i.

2.3.4 6-DOF Joint

The motion of two bodies not jointed together can be modeled as a six-degree-of-freedom *joint* that introduces no constraints. This is particularly useful for mobile robots, such as aircraft, that make at most intermittent contact with the ground, and thus, a body in free motion relative to the fixed frame is termed a *floating base*. Such a free motion joint model enables the position and orientation of a floating base in space to be expressed with six joint variables. The 6-DOF joint model is included in Table 2.6.

2.3.5 Physical Realization

In an actual robotic mechanism, the joints may have physical limits beyond which motion is prohibited. The workspace (Sect. 2.5) of a robotic manipulator is determined by considering the combined limits and freedom of motion of all the joints within the mechanism. Revolute joints are easily actuated by rotating motors and are, therefore, extremely common in robotic systems. They may also be present as passive, unactuated joints. Also common, although less so than revolutes, prismatic joints are relatively easily actuated by means of linear actuators such as hydraulic or pneumatic cylinders, ball screws, or screw jacks. They always have motion limits since unidirectional sliding can, in principle, produce infinite displacements. Helical joints are most often found in robotic mechanisms as constituents of linear actuators such as screw jacks and ball screws and are seldom used as primary kinematic joints. Joints with more than one degree of freedom are generally used passively in robotic mechanisms because each degree of freedom of an active joint must be separately actuated. Passive spherical joints are quite often found in robotic mechanisms, while passive planar joints are only occasionally found. The effect of an actuated spherical joint is achieved by employing the kinematically equivalent combination of three revolutes and actuating each. Universal joints are used in robotic mechanisms in both active and passive forms.

Serial chains are commonly denoted by the abbreviations for the joints they contain in the order in which they appear in the chain. For example, an RPR chain contains three links, the first jointed to the base with a revolute and to the second with a prismatic, while the second and third are jointed together with another revolute. If all of the joints are identical, the notation consists of the number of joints preceding the joint abbreviation, such as 6R for a six-axis serial-chain manipulator containing only revolute joints.

Joints are realized with hardware that is more complex than the idealizations presented in Sects. 2.3.1 and 2.3.2. For example, a revolute joint may be achieved with a ball bearing composed of a set of bearing balls trapped between two journals. The balls ideally roll without slipping on the journals, thereby taking advantage of the special properties of rolling contact joints. A prismatic joint may be realized by means of a roller-rail assembly.

2.3.6 Holonomic and Nonholonomic Constraints

With the exception of rolling contact, all of the constraints associated with the joints discussed in the preceding sections can be expressed mathematically by equations containing only the joint position variables. These are called holonomic constraints. The number of equations, and hence the number of constraints, is $6-n$, where n is the number of degrees of freedom of the joint. The constraints are intrinsically part of the axial joint model.

A nonholonomic constraint is one that cannot be expressed in terms of the position variables alone, but includes the time derivative of one or more of those variables. These constraint equations cannot be integrated to obtain relationships solely between the joint variables. The most common example in robotic systems arises from the use of a wheel or roller that rolls without slipping on another member. Nonholonomic constraints, particularly as they apply to wheeled robots, are discussed in more detail in Chap. 24.

2.3.7 Generalized Coordinates

In a robotic mechanism consisting of N bodies, $6N$ coordinates are required to specify the position and orientation of all the bodies relative to a coordinate frame. Since some of those bodies are jointed together, a number of constraint equations will establish relationships among some of these coordinates. In this case, the $6N$ coordinates can be expressed as functions of a smaller set of coordinates $\boldsymbol{q}$ that are all independent. The coordinates in this set are known as generalized coordinates, and motions associated with these coordinates are consistent with all of the constraints. The joint variables $\boldsymbol{q}$ of a robotic mechanism form a set of generalized coordinates [2.24, 25].

2.4 Geometric Representation

The geometry of a robotic mechanism is conveniently defined by attaching coordinate frames to each link. While these frames could be located arbitrarily, it is advantageous both for consistency and computational efficiency to adhere to a convention for locating the frames on the links. *Denavit* and *Hartenberg* [2.26] introduced the foundational convention that has been adapted in a number of different ways, one of which is the convention introduced by *Khalil* and *Dombre* [2.27] used throughout this handbook. In all of its forms, the convention requires only four rather than six parameters to locate one coordinate frame relative to another. The

four parameters consist of two link parameters, the link length a_i and the link twist α_i, and two joint parameters, the joint offset d_i and the joint angle θ_i. This parsimony is achieved through judicious placement of the coordinate frame origins and axes such that the $\hat{\boldsymbol{x}}$ axis of one frame both intersects and is perpendicular to the $\hat{\boldsymbol{z}}$ axis of the following coordinate frame. The convention is applicable to robotic mechanisms consisting of revolute and prismatic joints, so when multiple-degree-of-freedom joints are present, they are modeled as combinations of revolute and prismatic joints, as discussed in Sect. 2.3.

There are essentially four different forms of the convention for locating coordinate frames in a robotic mechanism. Each exhibits its own advantages by managing trade-offs of intuitive presentation. In the original *Denavit* and *Hartenberg* [2.26] convention, joint i is located between links i and $i+1$, so it is on the outboard side of link i. Also, the joint offset d_i and joint angle θ_i are measured along and about the $i-1$ joint axis, so the subscripts of the joint parameters do not match that of the joint axis. *Waldron* [2.28] and *Paul* [2.29] modified the labeling of axes in the original convention such that joint i is located between links $i-1$ and i in order to make it consistent with the base member of a serial chain being member 0. This places joint i at the inboard side of link i and is the convention used in all of the other modified versions. Furthermore, Waldron and Paul addressed the mismatch between subscripts of the joint parameters and joint axes by placing the $\hat{\boldsymbol{z}}_i$ axis along the $i+1$ joint axis. This, of course, relocates the subscript mismatch to the correspondence between the joint axis and the $\hat{\boldsymbol{z}}$ axis of the coordinate frame. *Craig* [2.30] eliminated all of the subscript mismatches by placing the $\hat{\boldsymbol{z}}_i$ axis along joint i, but at the expense of the homogeneous transformation ${}^{i-1}\mathbf{T}_i$ being formed with a mixture of joint parameters with subscript i and link parameters with subscript $i-1$. *Khalil* and *Dombre* [2.27] introduced another variation similar to Craig's except that it defines the link parameters a_i and α_i along and about the $\hat{\boldsymbol{x}}_{i-1}$ axis. In this case, the homogeneous transformation ${}^{i-1}\mathbf{T}_i$ is formed only by parameters with subscript i, and the subscript mismatch is such that a_i and α_i indicate the length and twist of link $i-1$ rather than link i. Thus, in summary, the advantages of the convention used throughout this handbook compared to the alternative conventions are that the $\hat{\boldsymbol{z}}$ axes of the coordinate frames share the common subscript of the joint axes and the four parameters that define the spatial transform from coordinate frame i to coordinate frame $i-1$ all share the common subscript i.

In this handbook, the convention for serial chain mechanisms is shown in Fig. 2.2 and summarized as follows. The numbering of bodies and joints follows the convention:

- The N moving bodies of the robotic mechanism are numbered from 1 to N. The number of the base is 0.
- The N joints of the robotic mechanism are numbered from 1 to N, with joint i located between members $i-1$ and i.

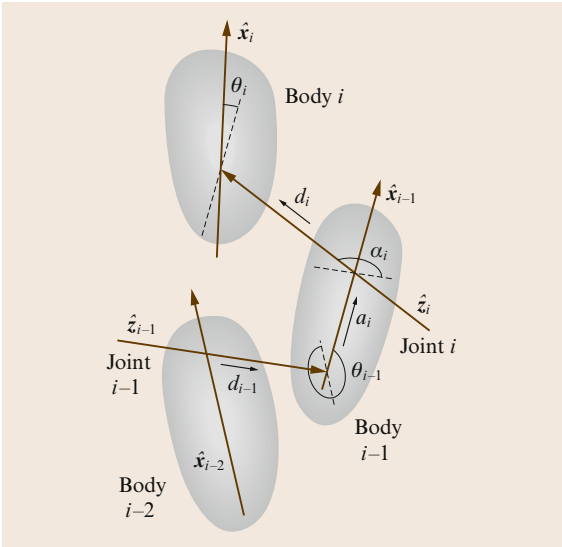


Fig. 2.2 Schematic of the numbering of bodies and joints in a robotic mechanism, the convention for attaching coordinate frames to the bodies, and the definitions of the four parameters, a_i, α_i, d_i, and θ_i, that locate one frame relative to another

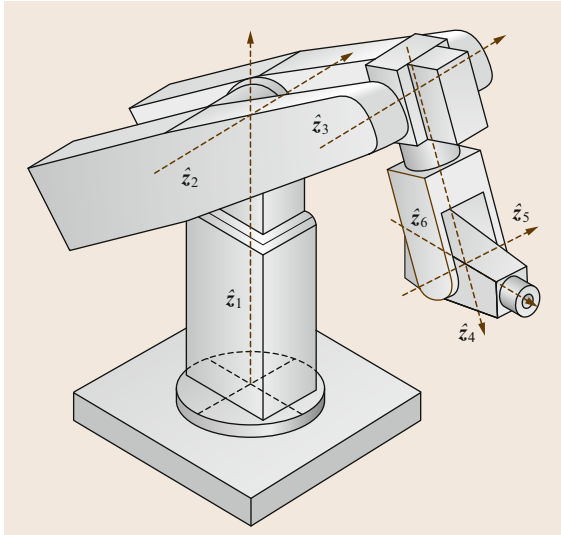


Fig. 2.3 Example six-degree-of-freedom serial chain manipulator composed of an articulated arm with no joint offsets and a spherical wrist

Table 2.7 Geometric parameters of the example serial chain manipulator in Fig. 2.3

i	α_i	a_i	d_i	θ_i
1	0	0	0	θ_1
2	$-\frac{\pi}{2}$	0	0	θ_2
3	0	a_3	0	θ_3
4	$-\frac{\pi}{2}$	0	d_4	θ_4
5	$\frac{\pi}{2}$	0	0	θ_5
6	$-\frac{\pi}{2}$	0	0	θ_6

With this numbering scheme, the attachment of coordinate frames follows the convention:

- The $\hat{\boldsymbol{z}}_i$ axis is located along the axis of joint i,
- The $\hat{\boldsymbol{x}}_{i-1}$ axis is located along the common normal between the $\hat{\boldsymbol{z}}_{i-1}$ and $\hat{\boldsymbol{z}}_i$ axes.

Using the attached frames, the four parameters that locate one frame relative to another are defined as:

- a_i is the distance from $\hat{\boldsymbol{z}}_{i-1}$ to $\hat{\boldsymbol{z}}_i$ along $\hat{\boldsymbol{x}}_{i-1}$,
- α_i is the angle from $\hat{\boldsymbol{z}}_{i-1}$ to $\hat{\boldsymbol{z}}_i$ about $\hat{\boldsymbol{x}}_{i-1}$,
- d_i is the distance from $\hat{\boldsymbol{x}}_{i-1}$ to $\hat{\boldsymbol{x}}_i$ along $\hat{\boldsymbol{z}}_i$,
- θ_i is the angle from $\hat{\boldsymbol{x}}_{i-1}$ to $\hat{\boldsymbol{x}}_i$ about $\hat{\boldsymbol{z}}_i$.

The geometric parameters for the example manipulator shown in Fig. 2.3 are listed in Table 2.7. All of the joints of this manipulator are revolutes, and joint 1 has a vertical orientation. Joint 2 is perpendicular to joint 1 and intersects it. Joint 3 is parallel to joint 2, and the length of link 2 is a_3. Joint 4 is perpendicular to joint 3 and intersects it. Joint 5 likewise intersects joint 4 perpendicularly at an offset of d_4 from joint 3. Finally, joint 6 intersects joint 5 perpendicularly.

With this convention, coordinate frame i can be located relative to coordinate frame $i-1$ by executing a rotation through an angle α_i about the $\hat{\boldsymbol{x}}_{i-1}$ axis, a translation of distance a_i along $\hat{\boldsymbol{x}}_{i-1}$, a rotation through an angle θ_i about the $\hat{\boldsymbol{z}}_i$ axis, and a translation of distance d_i along $\hat{\boldsymbol{z}}_i$. Through concatenation of these individual transformations,

$$\mathbf{Rot}(\hat{\boldsymbol{x}}_{i-1}, \alpha_i)\mathbf{Trans}(\hat{\boldsymbol{x}}_{i-1}, a_i)\mathbf{Rot}(\hat{\boldsymbol{z}}_i, \theta_i)\mathbf{Trans}(\hat{\boldsymbol{z}}_i, d_i) \,, \tag{2.43}$$

the equivalent homogeneous transformation is,

$$^{i-1}\mathbf{T}_i = \begin{pmatrix} \cos\theta_i & -\sin\theta_i & 0 & a_i \\ \sin\theta_i\cos\alpha_i & \cos\theta_i\cos\alpha_i & -\sin\alpha_i & -\sin\alpha_i d_i \\ \sin\theta_i\sin\alpha_i & \cos\theta_i\sin\alpha_i & \cos\alpha_i & \cos\alpha_i d_i \\ 0 & 0 & 0 & 1 \end{pmatrix} . \tag{2.44}$$

The identification of geometric parameters is addressed in Chap. 8.

2.5 Workspace

Most generally, the workspace of a robotic manipulator is the total volume swept out by the end-effector as the manipulator executes all possible motions. The workspace is determined by the geometry of the manipulator and the limits of the joint motions. It is more specific to define the reachable workspace as the total locus of points at which the end-effector can be placed and the dextrous workspace [2.31] as the subset of those points at which the end-effector can be placed while having an arbitrary orientation. Dexterous workspaces exist only for certain idealized geometries, so real industrial manipulators with joint motion limits almost never possess dexterous workspaces.

Many serial-chain robotic manipulators are designed such that their joints can be partitioned into a regional structure and an orientation structure. The joints in the regional structure accomplish the positioning of the end-effector in space, and the joints in the orientation structure accomplish the orientation of the end-effector. Typically, the inboard joints of a serial chain manipulator comprise the regional structure, while the outboard joints comprise the orientation structure. Also, since prismatic joints provide no capability for rotation, they are generally not employed within the orientation structure.

The regional workspace volume can be calculated from the known geometry of the serial-chain manipulator and motion limits of the joints. With three inboard joints comprising the regional structure, the area of workspace for the outer two (joints 2 and 3) is computed first, and then the volume is calculated by integrating over the joint variable of the remaining inboard joint (joint 1). In the case of a prismatic joint, this simply involves multiplying the area by the total length of travel of the prismatic joint. In the more common case of a revolute joint, it involves rotating the area about the joint axis through the full range of motion of the revolute [2.32]. By the theorem of Pappus, the associated volume V is

$$V = A\bar{r}\gamma \,, \tag{2.45}$$

where A is the area, $\bar{r}$ is the distance from the area's centroid to the axis, and γ is the angle through which the area is rotated. The boundaries of the area are determined by tracing the motion of a reference point in the end-effector, typically the center of rotation of the wrist that serves as the orientation structure. Starting with each of the two joints at motion limits and with joint 2 locked, joint 3 is moved until its second motion limit is reached. Joint 3 is then locked, and joint 2 is freed to move to its second motion limit. Joint 2 is again locked, while joint 3 is freed to move back to its original motion limit. Finally, joint 3 is locked, and joint 2 freed to move likewise to its original motion limit. In this way, the trace of the reference point is a closed curve whose area and centroid can be calculated mathematically.

More details on manipulator workspace can be found in Chaps. 4 and 16.

2.6 Forward Kinematics

The forward kinematics problem for a serial-chain manipulator is to find the position and orientation of the end-effector relative to the base given the positions of all of the joints and the values of all of the geometric link parameters. Often, a frame fixed in the end-effector is referred to as the *tool frame*, and while fixed in the final link N, it in general has a constant offset in both position and orientation from frame N. Likewise, a *station frame* is often located in the base to establish the location of the task to be performed. This frame generally has a constant offset in its pose relative to frame 0, which is also fixed in the base.

A more general expression of the forward kinematics problem is to find the relative position and orientation of any two designated members given the geometric structure of the robotic mechanism and the values of a number of joint positions equal to the number of degrees of freedom of the mechanism. The forward kinematics problem is critical for developing manipulator coordination algorithms because joint positions are typically measured by sensors mounted on the joints and it is necessary to calculate the positions of the joint axes relative to the fixed frame.

In practice, the forward kinematics problem is solved by calculating the transformation between a coordinate frame fixed in the end-effector and another coordinate frame fixed in the base, i. e., between the tool and station frames. This is straightforward for a serial chain since the transformation describing the position of the end-effector relative to the base is obtained by simply concatenating transformations between frames fixed in adjacent links of the chain. The convention for the geometric representation of a manipulator presented in Sect. 2.4 reduces this to finding an equivalent 4×4 homogeneous transformation matrix that relates the spatial displacement of the end-effector coordinate frame to the base frame.

For the example serial-chain manipulator shown in Fig. 2.3 and neglecting the addition of tool and station

Table 2.8 Forward kinematics of the example serial chain manipulator in Fig. 2.3, with abbreviations $c_{\theta_i} := \cos\theta_i$ and $s_{\theta_i} := \sin\theta_i$

$$
{}^0\mathbf{T}_6 = \begin{pmatrix} r_{11} & r_{12} & r_{13} & {}^0p_6^x \\ r_{21} & r_{22} & r_{23} & {}^0p_6^y \\ r_{31} & r_{32} & r_{33} & {}^0p_6^z \\ 0 & 0 & 0 & 1 \end{pmatrix},
$$

$$
\begin{aligned}
r_{11} &= c_{\theta_1}(s_{\theta_2}s_{\theta_3} - c_{\theta_2}c_{\theta_3})(s_{\theta_4}s_{\theta_6} - c_{\theta_4}c_{\theta_5}c_{\theta_6}) \\
&\quad - c_{\theta_1}s_{\theta_5}c_{\theta_6}(c_{\theta_2}s_{\theta_3} + s_{\theta_2}c_{\theta_3}) \\
&\quad + s_{\theta_1}(s_{\theta_4}c_{\theta_5}c_{\theta_6} + c_{\theta_4}s_{\theta_6}) , \\
r_{21} &= s_{\theta_1}(s_{\theta_2}s_{\theta_3} - c_{\theta_2}c_{\theta_3})(s_{\theta_4}s_{\theta_6} - c_{\theta_4}c_{\theta_5}c_{\theta_6}) \\
&\quad - s_{\theta_1}s_{\theta_5}c_{\theta_6}(c_{\theta_2}s_{\theta_3} + s_{\theta_2}c_{\theta_3}) \\
&\quad - c_{\theta_1}(s_{\theta_4}c_{\theta_5}c_{\theta_6} + c_{\theta_4}s_{\theta_6}) , \\
r_{31} &= (c_{\theta_2}s_{\theta_3} + s_{\theta_2}c_{\theta_3})(s_{\theta_4}s_{\theta_6} - c_{\theta_4}c_{\theta_5}c_{\theta_6}) \\
&\quad + s_{\theta_5}c_{\theta_6}(s_{\theta_2}s_{\theta_3} - c_{\theta_2}c_{\theta_3}) , \\
r_{12} &= c_{\theta_1}(s_{\theta_2}s_{\theta_3} - c_{\theta_2}c_{\theta_3})(c_{\theta_4}c_{\theta_5}s_{\theta_6} + s_{\theta_4}c_{\theta_6}) \\
&\quad + c_{\theta_1}s_{\theta_5}s_{\theta_6}(c_{\theta_2}s_{\theta_3} + s_{\theta_2}c_{\theta_3}) \\
&\quad + s_{\theta_1}(c_{\theta_4}c_{\theta_6} - s_{\theta_4}c_{\theta_5}s_{\theta_6}) , \\
r_{22} &= s_{\theta_1}(s_{\theta_2}s_{\theta_3} - c_{\theta_2}c_{\theta_3})(c_{\theta_4}c_{\theta_5}s_{\theta_6} + s_{\theta_4}c_{\theta_6}) \\
&\quad + s_{\theta_1}s_{\theta_5}s_{\theta_6}(c_{\theta_2}s_{\theta_3} + s_{\theta_2}c_{\theta_3}) \\
&\quad - c_{\theta_1}(c_{\theta_4}c_{\theta_6} - s_{\theta_4}c_{\theta_5}s_{\theta_6}) , \\
r_{32} &= (c_{\theta_2}s_{\theta_3} + s_{\theta_2}c_{\theta_3})(c_{\theta_4}c_{\theta_5}s_{\theta_6} + s_{\theta_4}c_{\theta_6}) \\
&\quad - s_{\theta_5}s_{\theta_6}(s_{\theta_2}s_{\theta_3} - c_{\theta_2}c_{\theta_3}) , \\
r_{13} &= c_{\theta_1}c_{\theta_4}s_{\theta_5}(s_{\theta_2}s_{\theta_3} - c_{\theta_2}c_{\theta_3}) \\
&\quad - c_{\theta_1}c_{\theta_5}(c_{\theta_2}s_{\theta_3} + s_{\theta_2}c_{\theta_3}) \\
&\quad - s_{\theta_1}s_{\theta_4}s_{\theta_5} , \\
r_{23} &= s_{\theta_1}c_{\theta_4}s_{\theta_5}(s_{\theta_2}s_{\theta_3} - c_{\theta_2}c_{\theta_3}) \\
&\quad - s_{\theta_1}c_{\theta_5}(c_{\theta_2}s_{\theta_3} + s_{\theta_2}c_{\theta_3}) + c_{\theta_1}s_{\theta_4}s_{\theta_5} , \\
r_{33} &= c_{\theta_4}s_{\theta_5}(c_{\theta_2}s_{\theta_3} + s_{\theta_2}c_{\theta_3}) \\
&\quad + c_{\theta_5}(s_{\theta_2}s_{\theta_3} - c_{\theta_2}c_{\theta_3}) , \\
{}^0p_6^x &= a_3c_{\theta_1}c_{\theta_2} - d_4c_{\theta_1}(c_{\theta_2}s_{\theta_3} + s_{\theta_2}c_{\theta_3}) , \\
{}^0p_6^y &= a_3s_{\theta_1}c_{\theta_2} - d_4s_{\theta_1}(c_{\theta_2}s_{\theta_3} + s_{\theta_2}c_{\theta_3}) , \\
{}^0p_6^z &= -a_3s_{\theta_2} + d_4(s_{\theta_2}s_{\theta_3} - c_{\theta_2}c_{\theta_3}) .
\end{aligned}
$$

frames, the transformation is

$$ {}^0\mathbf{T}_6 = {}^0\mathbf{T}_1\,{}^1\mathbf{T}_2\,{}^2\mathbf{T}_3\,{}^3\mathbf{T}_4\,{}^4\mathbf{T}_5\,{}^5\mathbf{T}_6 \,. \tag{2.46} $$

Table 2.8 contains the elements of ${}^0\mathbf{T}_6$ that are calculated using Table 2.7 and (2.44).

Once again, homogeneous transformations provide a compact notation, but are computationally inefficient for solving the forward kinematics problem. A reduction in computation can be achieved by separating the position and orientation portions of the transformation to eliminate all multiplications by the 0 and 1 elements of the matrices. In Chap. 3, calculations are made using the spatial vector notation briefly introduced here in Sect. 2.2.6 and explained in detail in Sect. 3.2. This approach does not employ homogeneous transformations, but rather separates out the rotation matrices and positions to achieve computation efficiency. Table 3.1 provides the detailed formulas, with the product of spatial transforms particularly relevant to the forward kinematics problem.

Kinematic trees are the general structure of robotic mechanisms that do not contain closed loops, and the forward kinematics of tree structures are addressed in Chap. 3. The forward kinematics problem for closed chains is much more complicated because of the additional constraints present. Solution methods for closed chains are included in Chap. 18.

2.7 Inverse Kinematics

The inverse kinematics problem for a serial-chain manipulator is to find the values of the joint positions given the position and orientation of the end-effector relative to the base and the values of all of the geometric link parameters. Once again, this is a simplified statement applying only to serial chains. A more general statement is: given the relative positions and orientations of two members of a mechanism, find the values of all of the joint positions. This amounts to finding all of the joint positions given the homogeneous transformation between the two members of interest.

In the common case of a six-degree-of-freedom serial chain manipulator, the known transformation is ${}^0\mathbf{T}_6$. Reviewing the formulation of this transformation in Sect. 2.6, it is clear that the inverse kinematics problem for serial-chain manipulators requires the solution of sets of nonlinear equations. In the case of a six-degree-of-freedom manipulator, three of these equations relate to the position vector within the homogeneous transformation, and the other three relate to the rotation matrix. In the latter case, these three equations cannot come from the same row or column because of the dependency within the rotation matrix. With these nonlinear equations, it is possible that no solutions exist or multiple solutions exist [2.33]. For a solution to exist, the desired position and orientation of the end-effector must lie in the workspace of the manipulator. In cases where solutions do exist, they often cannot be presented in closed form, so numerical methods are required.

2.7.1 Closed-Form Solutions

Closed-form solutions are desirable because they are faster than numerical solutions and readily identify all possible solutions. The disadvantage of closed-form solutions is that they are not general, but robot dependent. The most effective methods for finding closed-form solutions are ad hoc techniques that take advantage of particular geometric features of specific mechanisms. In general, closed-form solutions can only be obtained for six-degree-of-freedom systems with special kinematic structure characterized by a large number of the geometric parameters defined in Sect. 2.4 being zero-valued. Most industrial manipulators have such structure because it permits more efficient coordination software. Sufficient conditions for a six-degree-of-freedom manipulator to have closed-form inverse kinematics solutions are [2.34–36]:

1. Three consecutive revolute joint axes intersect at a common point, as in a spherical wrist.
2. Three consecutive revolute joint axes are parallel.

Closed-form solution approaches are generally divided into algebraic and geometric methods.

Algebraic Methods

Algebraic methods involve identifying the significant equations containing the joint variables and manipulating them into a soluble form. A common strategy is reduction to a transcendental equation in a single variable such as,

$$ C_1 \cos\theta_i + C_2 \sin\theta_i + C_3 = 0 \,, \tag{2.47} $$

where C_1, C_2, and C_3 are constants. The solution to such an equation is

$$ \theta_i = 2\tan^{-1}\left(\frac{C_2 \pm \sqrt{C_2^2 - C_3^2 + C_1^2}}{C_1 - C_3}\right) . \tag{2.48} $$

Special cases in which one or more of the constants are zero are also common.

Reduction to a pair of equations having the form,

$$C_1 \cos\theta_i + C_2 \sin\theta_i + C_3 = 0\,, \tag{2.49}$$

$$C_1 \sin\theta_i - C_2 \cos\theta_i + C_4 = 0\,, \tag{2.50}$$

is another particularly useful strategy because only one solution results,

$$\theta_i = \mathrm{Atan2}(-C_1C_4 - C_2C_3, C_2C_4 - C_1C_3)\,. \tag{2.51}$$

Geometric Methods

Geometric methods involve identifying points on the manipulator relative to which position and/or orientation can be expressed as a function of a reduced set of the joint variables. This often amounts to decomposing the spatial problem into separate planar problems. The resulting equations are solved using algebraic manipulation. The two sufficient conditions for existence of a closed-form solution for a six-degree-of-freedom manipulator that are listed above enable the decomposition of the problem into inverse position kinematics and inverse orientation kinematics. This is the decomposition into regional and orientation structures discussed in Sect. 2.5, and the solution is found by rewriting (2.46),

$${}^0\mathbf{T}_6\,{}^6\mathbf{T}_5\,{}^5\mathbf{T}_4\,{}^4\mathbf{T}_3 = {}^0\mathbf{T}_1\,{}^1\mathbf{T}_2\,{}^2\mathbf{T}_3\,. \tag{2.52}$$

The example manipulator in Fig. 2.3 has this structure, and its regional structure is commonly known as an articulated or anthropomorphic arm or an elbow manipulator. The solution to the inverse position kinematics problem for such a structure is summarized in Table 2.9. Because there are two solutions for θ_1 and likewise two solutions for both θ_2 and θ_3 corresponding to each θ_1 solution, there are a total of four solutions to the inverse position kinematics problem of the articulated arm manipulator. The orientation structure is simply a spherical wrist, and the corresponding solution to the inverse orientation kinematics problem is summarized in Table 2.10. Two solutions for θ_5 are given in Table 2.10, but only one solution for both θ_4 and θ_6 corresponds to each. Thus, the inverse orientation kinematics problem of a spherical wrist has two solutions. Combining the regional and orientation structures, the total number of inverse kinematics solutions for the manipulator in Fig. 2.3 is eight.

2.7.2 Numerical Methods

Unlike the algebraic and geometric methods used to find closed-form solutions, numerical methods are not robot dependent, so they can be applied to any kinematic structure. The disadvantages of numerical methods are that they can be slower and in some cases, they do not allow computation of all possible solutions. For a six-degree-of-freedom serial-chain manipulator with only revolute and prismatic joints, the translation and rotation equations can always be reduced to a polynomial in a single variable of degree not greater than 16 [2.37]. Thus, such a manipulator can have as many as 16 real solutions to the inverse kinematics problem [2.38]. Since closed-form solution of a polynomial equation is only possible if the polynomial is of degree four or less, it follows that many manipulator geometries are not soluble in closed form. In general, a greater number of nonzero geometric parameters corresponds to a polynomial of higher degree in the reduction. For such manipulator structures, the most common numerical methods can be divided into categories of symbolic elimination methods, continuation methods, and iterative methods.

Symbolic Elimination Methods

Symbolic elimination methods involve analytical manipulations to eliminate variables from the system of nonlinear equations to reduce it to a smaller set of equations. *Raghavan* and *Roth* [2.39] used dialytic elimination to reduce the inverse kinematics problem of a general 6R serial-chain manipulator to a polyno-

Table 2.9 Inverse position kinematics of the articulated arm within the example serial chain manipulator in Fig. 2.3

$\theta_1 = \mathrm{Atan2}\left({}^0p_6^y, {}^0p_6^x\right)$ or $\mathrm{Atan2}\left(-{}^0p_6^y, -{}^0p_6^x\right)$
$\theta_3 = -\mathrm{Atan2}\left(D, \pm\sqrt{1-D^2}\right)$, where $D := \frac{({}^0p_6^x)^2+({}^0p_6^y)^2+({}^0p_6^z)^2-a_3^2-d_4^2}{2a_3d_4}$,
$\theta_2 = \mathrm{Atan2}\left({}^0p_6^z, \sqrt{({}^0p_6^x)^2+({}^0p_6^y)^2}\right)$ $-\mathrm{Atan2}\left(d_4\cos\theta_3, a_3 - d_4\sin\theta_3\right)$

Table 2.10 Inverse orientation kinematics of the spherical wrist within the example serial chain manipulator in Fig. 2.3, with abbreviations $c_{\theta_i} := \cos\theta_i$ and $s_{\theta_i} := \sin\theta_i$

$\theta_5 = \mathrm{Atan2}\left(\pm\sqrt{1-\left(r_{13}s_{\theta_1} - r_{23}c_{\theta_1}\right)^2}\,, r_{13}s_{\theta_1} - r_{23}c_{\theta_1}\right)$
$\theta_4 = \mathrm{Atan2}\left(\mp\left(r_{13}c_{\theta_1} + r_{23}s_{\theta_1}\right)s_{(\theta_2+\theta_3)} \mp r_{33}c_{(\theta_2+\theta_3)}\,,\right.$ $\left.\pm\left(r_{13}c_{\theta_1} + r_{23}s_{\theta_1}\right)c_{(\theta_2+\theta_3)} \mp r_{23}s_{(\theta_2+\theta_3)}\right)$
$\theta_6 = \mathrm{Atan2}\left(\pm\left(r_{12}s_{\theta_1} + r_{22}c_{\theta_1}\right), \pm\left(r_{11}s_{\theta_1} - r_{21}c_{\theta_1}\right)\right)$, where the $\pm$ choice for θ_5 dictates all of the subsequent $\pm$ and $\mp$ for θ_4 and θ_6 .

mial of degree 16 and to find all possible solutions. The roots provide solutions for one of the joint variables, while the other variables are computed by solving linear systems. *Manocha* and *Canny* [2.40] improved the numerical properties of this technique by reformulating the problem as a generalized eigenvalue problem. An alternative approach to elimination makes use of Gröbner bases [2.41, 42].

Continuation Methods

Continuation methods involve tracking a solution path from a start system with known solutions to a target system whose solutions are sought as the start system is transformed into the target system. These techniques have been applied to inverse kinematics problems [2.43], and special properties of polynomial systems can be exploited to find all possible solutions [2.44].

Iterative Methods

A number of different iterative methods can be employed to solve the inverse kinematics problem. Most of them converge to a single solution based on an initial guess, so the quality of that guess greatly impacts the solution time. Newton–Raphson methods provide a fundamental approach that uses a first-order approximation of the original equations. *Pieper* [2.34] was among the first to apply the method to inverse kinematics, and others have followed [2.45, 46]. Optimization approaches formulate the problem as a nonlinear optimization problem and employ search techniques to move from an initial guess to a solution [2.47, 48]. Resolved motion rate control converts the problem to a differential equation [2.49], and a modified predictor–corrector algorithm can be used to perform the joint velocity integration [2.50]. Control-theory-based methods cast the differential equation into a control problem [2.51]. Interval analysis [2.52] is perhaps one of the most promising iterative methods because it offers rapid convergence to a solution and can be used to find all possible solutions. For complex mechanisms, the damped least-squares approach [2.53] is particularly attractive, and more detail is provided in Chap. 10.

2.8 Forward Instantaneous Kinematics

The forward instantaneous kinematics problem for a serial-chain manipulator is: given the positions of all members of the chain and the rates of motion about all the joints, find the total velocity of the end-effector. Here the rate of motion about the joint is the angular velocity of rotation about a revolute joint or the translational velocity of sliding along a prismatic joint. The total velocity of a member is the velocity of the origin of the coordinate frame fixed to it combined with its angular velocity. That is, the total velocity has six independent components and therefore, completely represents the velocity field of the member. It is important to note that this problem definition includes an assumption that the pose of the mechanism is completely known. In most situations, this means that either the forward or inverse position kinematics problem must be solved before the forward instantaneous kinematics problem can be addressed. The same is true of the inverse instantaneous kinematics problem discussed in the following section. The forward instantaneous kinematics problem is important when doing acceleration analysis for the purpose of studying dynamics. The total velocities of the members are needed for the computation of Coriolis and centripetal acceleration components.

2.8.1 Jacobian

Differentiation with respect to time of the forward position kinematics equations yields a set of equations of the form

$$^{k}\boldsymbol{v}_{N} = \mathbf{J}(\boldsymbol{q})\dot{\boldsymbol{q}}, \tag{2.53}$$

where $^{k}\boldsymbol{v}_{N}$ is the spatial velocity of the end-effector expressed in any frame k, $\dot{\boldsymbol{q}}$ is an n-dimensional vector composed of the joint rates, and $\mathbf{J}(\boldsymbol{q})$ is a $6\times n$ matrix whose elements are, in general, nonlinear functions of $\boldsymbol{q}$. $\mathbf{J}(\boldsymbol{q})$ is called the Jacobian matrix of this algebraic system and is expressed relative to the same coordinate frame as the spatial velocity $^{k}\boldsymbol{v}_{N}$ [2.54]. Alternately, (2.53) can be expressed as

$$^{k}\boldsymbol{v}_{N} = [\mathbf{J}_1 \;\; \mathbf{J}_2 \;\; \cdots \;\; \mathbf{J}_N]\;\dot{\boldsymbol{q}}\,, \tag{2.54}$$

where N is the number of joints (each with possibly more than 1 degree-of-freedom) and $\mathbf{J}_i$ provides the column(s) of $\mathbf{J}(\boldsymbol{q})$ which correspond(s) to $\dot{\boldsymbol{q}}_i$. If the joint positions are known, (2.53) yields six linear algebraic equations in the joint rates. If the joint rates are given,

a solution of (2.53) is a solution of the forward instantaneous kinematics problem. Note that $\mathbf{J}(\boldsymbol{q})$ can be regarded as a known matrix for this purpose provided all the joint positions are known.

Algorithm 2.1 Jacobian Computation Algorithm for a Serial-Chain Mechanism

inputs: ${}^{k}\mathbf{X}_N$, ${}^{N}\mathbf{X}_{N-1}, \ldots, {}^{i}\mathbf{X}_{i-1}, \ldots, {}^{2}\mathbf{X}_1$
output: $\mathbf{J}$

$\mathbf{X} = {}^{k}\mathbf{X}_N$
for $i = N$ **to** 1 **do**
  $\mathbf{J}_i = \mathbf{X}\,\boldsymbol{\Phi}_i$
  if $i > 1$ **then**
    $\mathbf{X} = \mathbf{X}\,{}^{i}\mathbf{X}_{i-1}$
  end if
end for

Using the spatial vector notation briefly introduced in Sect. 2.2.6 and explained in detail in Sect. 3.2, the Jacobian can be easily computed from the free modes $\boldsymbol{\Phi}_i$ of the joints and the associated spatial transforms ${}^{j}\mathbf{X}_i$.

$$\mathbf{J}_i = {}^{k}\mathbf{X}_i\,\boldsymbol{\Phi}_i\,. \tag{2.55}$$

To understand why this holds, note that $\boldsymbol{\Phi}_i$ describes the spatial velocities created by joint i in local coordinates and ${}^{k}\mathbf{X}_i$ transforms spatial velocities expressed in frame i to frame k. The quantities $\mathbf{J}_i$ can be efficiently computed from ${}^{k}\mathbf{X}_N$ and the link-to-link transforms ${}^{i}\mathbf{X}_{i-1}$. Algorithm 2.1 contains an algorithm for computing the columns of the Jacobian in this manner. Table 3.1 provides efficient methods to carry out the multiplications required in this algorithm which exploit the structure of spatial transformation matrices. Note that the quantity ${}^{k}\mathbf{X}_N$ can be computed with forward kinematics and each ${}^{i}\mathbf{X}_{i-1}$ can be computed with simple joint kinematics. Thus, use of the algorithm in Alg. 2.1 simplifies the problem of forward instantaneous kinematics to one of forward standard kinematics. Additional information about the Jacobian can be found in Chap. 10.

2.9 Inverse Instantaneous Kinematics

The important problem from the point of view of robotic coordination is the inverse instantaneous kinematics problem. More information on robot coordination can be found in Chaps. 7 and 8. The inverse instantaneous kinematics problem for a serial chain manipulator is: given the positions of all members of the chain and the total velocity of the end-effector, find the rates of motion of all joints. When controlling a movement of an industrial robot that operates in the point-to-point mode, it is not only necessary to compute the final joint positions needed to assume the desired final hand position. It is also necessary to generate a smooth trajectory for motion between the initial and final positions. There are, of course, an infinite number of possible trajectories for this purpose. However, the most straightforward and successful approach employs algorithms based on the solution of the inverse instantaneous kinematics problem. This technique originated in the work of *Whitney* [2.55] and of *Pieper* [2.34].

2.9.1 Inverse Jacobian

In order to solve the linear system of equations in the joint rates obtained by decomposing (2.53) into its component equations when $\mathbf{v}_N$ is known, it is necessary to invert the Jacobian matrix. The equation becomes

$$\dot{\boldsymbol{q}} = \mathbf{J}^{-1}(\boldsymbol{q})\mathbf{v}_N\,. \tag{2.56}$$

Since $\mathbf{J}$ is a 6×6 matrix, numerical inversion is not very attractive. It is quite possible for $\mathbf{J}$ to become singular ($|\mathbf{J}| = 0$), in which case the inverse does not exist. More information on singularities can be found in Chaps. 4 and 18. Even when the Jacobian matrix does not become singular, it may become ill-conditioned, leading to degraded performance in significant portions of the manipulator's workspace. Most industrial robot geometries are simple enough that the Jacobian matrix can be inverted analytically, leading to a set of explicit equations for the joint rates [2.56–58]. This greatly reduces the number of operations needed as compared to numerical inversion. For more complex manipulator geometries, though, numerical inversion is the only solution option. The Jacobian of a redundant manipulator is not square, so it cannot be inverted. Chapter 10 discusses how various pseudoinverses can be used in such cases.

2.10 Static Wrench Transmission

A general force system can be shown to be equivalent to a single force together with a moment acting about its line of action. This is called a wrench. There is a deep isometry between the geometries of systems of wrench axes and that of systems of instantaneous screw axes [2.59]. Static wrench analysis of a manipulator establishes the relationship between wrenches applied to the end-effector and forces/torques applied to the joints. This is essential for controlling a manipulator's interactions with its environment. Examples include tasks involving fixed or quasi-fixed workpieces such as inserting a component in place with a specified force and tightening a nut to a prescribed torque. More information can be found in Chaps. 9 and 37. Through the principle of virtual work, the relationship between wrenches applied to the end-effector and forces/torques applied to the joints can be shown to be

$$\boldsymbol{\tau} = \mathbf{J}^{\mathrm{T}}\mathbf{f}\,, \tag{2.57}$$

where $\boldsymbol{\tau}$ is the n-dimensional vector of applied joint forces/torques for an n-degree-of-freedom manipulator and $\boldsymbol{f}$ is the spatial force vector

$$\mathbf{f} = \begin{pmatrix} \boldsymbol{n} \\ \boldsymbol{f} \end{pmatrix}\,, \tag{2.58}$$

in which $\boldsymbol{n}$ and $\boldsymbol{f}$ are the vectors of torques and forces, respectively, applied to the end-effector, both expressed in the coordinate frame relative to which the Jacobian is also expressed. Thus, in the same way the Jacobian maps the joint rates to the spatial velocity of the end-effector, its transpose maps the wrenches applied to the end-effector to the equivalent joint forces/torques. As in the velocity case, when the Jacobian is not square, the inverse relationship is not uniquely defined.

2.11 Conclusions and Further Reading

This chapter presents an overview of how the fundamentals of kinematics can be applied to robotic mechanisms. The topics include various representations of the position and orientation of a rigid body in space, the freedom of motion and accompanying mathematical models of joints, a geometric representation that describes the bodies and joints of a robotic mechanism, the workspace of a manipulator, the problems of forward and inverse kinematics, the problems of forward and inverse instantaneous kinematics including the definition of the Jacobian, and finally the transmission of static wrenches. This chapter is certainly not a comprehensive account of robot kinematics. Fortunately, a number of excellent texts provide a broad introduction to robotics with significant focus on kinematics [2.17, 27, 29, 30, 51, 60–64].

From a historical perspective, robotics fundamentally changed the nature of the field of mechanism kinematics. Before the first work on the generation of coordination equations for robots [2.34, 55], the focus of the field was almost entirely on single-degree-of-freedom mechanisms. This is why robotics, following on from the advent of digital computing, led to a renaissance of work in mechanism kinematics. More details can be found in Chap. 4. The evolution of the field has continued as it has broadened from the study of simple serial chains for industrial robots, the focus of the analysis in this chapter, to parallel machines (Chap. 18), human-like grippers (Chap. 19), robotic vehicles (Chaps. 17 and 24–26), and even small-scale robots (Chap. 27).

References

2.1 W. R. Hamilton: On quaternions, or on a new system of imaginaries in algebra, Philos. Mag. **18** (2000)

2.2 E.B. Wilson: *Vector Analysis* (Dover, New York 1960), based upon the lectures of J.W. Gibbs (reprint of the 2nd edn. published by Charles Scribner's Sons, 1909)

2.3 H. Graßmann: *Die Wissenschaft der extensiven Größe oder die Ausdehnungslehre* (Wigand, Leipzig 1844)

2.4 J.M. McCarthy: *Introduction to Theoretical Kinematics* (MIT Press, Cambridge 1990)

2.5 W.K. Clifford: Preliminary sketch of bi-quarternions, Proc. Lond. Math. Soc. **4**, 381–395 (1873)

2.6 A.P. Kotelnikov: *Screw calculus and some applications to geometry and mechanics* (Annal. Imp. Univ., Kazan 1895)

2.7 E. Study: *Geometrie der Dynamen* (Teubner, Leipzig 1903)

2.8 G.S. Chirikjian, A.B. Kyatkin: *Engineering Applications of Noncommutative Harmonic Analysis* (CRC, Boca Raton 2001)

2.9 R. von Mises: Anwendungen der Motorrechnung, Z. Angew. Math. Mech. **4**(3), 193–213 (1924)

2.10 J.E. Baker, I.A. Parkin: *Fundamentals of Screw Motion: Seminal Papers by Michel Chasles and Olinde Rodrigues*, School of Information Technologies (University of Sydney, Sydney 2003), translated from O. Rodrigues: Des lois géométriques qui régissent les déplacements d'un système dans l'espace, J. Math. Pures Applicqu. Liouville 5, 380–440 (1840)

2.11 R.S. Ball: *A Treatise on the Theory of Screws* (Cambridge Univ. Press, Cambridge 1998)

2.12 J.K. Davidson, K.H. Hunt: *Robots and Screw Theory: Applications of Kinematics and Statics to Robotics* (Oxford Univ. Press, Oxford 2004)

2.13 K.H. Hunt: *Kinematic Geometry of Mechanisms* (Clarendon, Oxford 1978)

2.14 J.R. Phillips: *Freedom in Machinery. Vol 1. Introducing Screw Theory* (Cambridge Univ. Press, Cambridge 1984)

2.15 J.R. Phillips: *Freedom in Machinery. Vol 2. Screw Theory Exemplified* (Cambridge Univ. Press, Cambridge 1990)

2.16 G.S. Chirikjian: Rigid-body kinematics. In: *Robotics and Automation Handbook*, ed. by T. Kurfess (CRC, Boca Raton 2005), Chap. 2

2.17 R.M. Murray, Z. Li, S.S. Sastry: *A Mathematical Introduction to Robotic Manipulation* (CRC, Boca Raton 1994)

2.18 A. Karger, J. Novak: *Space Kinematics and Lie Groups* (Routledge, New York 1985)

2.19 R. von Mises: Motorrechnung, ein neues Hilfsmittel in der Mechanik, Z. Angew. Math. Mech. **2**(2), 155–181 (1924)

2.20 J.D. Everett: On a new method in statics and kinematics, Mess. Math. **45**, 36–37 (1875)

2.21 F. Reuleaux: *Kinematics of Machinery* (Dover, New York 1963), reprint of *Theoretische Kinematik*, 1875, in German

2.22 R. Featherstone: *Rigid Body Dynamics Algorithms* (Kluwer, Boston 2007)

2.23 K.J. Waldron: A method of studying joint geometry, Mechan. Mach. Theory **7**, 347–353 (1972)

2.24 T.R. Kane, D.A. Levinson: *Dynamics, Theory and Applications* (McGraw-Hill, New York 1985)

2.25 J.L. Lagrange: *Oeuvres de Lagrange* (Gauthier-Villars, Paris 1867)

2.26 J. Denavit, R.S. Hartenberg: A kinematic notation for lower-pair mechanisms based on matrices, J. Appl. Mech. **22**, 215–221 (1955)

2.27 W. Khalil, E. Dombre: *Modeling, Identification and Control of Robots* (Taylor Francis, New York 2002)

2.28 K.J. Waldron: A study of overconstrained linkage geometry by solution of closure equations, Part I: A method of study, Mech. Mach. Theory **8**(1), 95–104 (1973)

2.29 R. Paul: *Robot Manipulators: Mathematics, Programming and Control* (MIT Press, Cambridge 1982)

2.30 J.J. Craig: *Introduction to Robotics: Mechanics and Control* (Addison-Wesley, Reading 1986)

2.31 K.J. Waldron, A. Kumar: The dextrous workspace, ASME Mech. Conf., Los Angeles (1980), ASME paper No. 80-DETC-108

2.32 R. Vijaykumar, K.J. Waldron, M.J. Tsai: Geometric optimization of manipulator structures for working volume and dexterity, Int. J. Robotics Res. **5**(2), 91–103 (1986)

2.33 J. Duffy: *Analysis of Mechanisms and Robot Manipulators* (Wiley, New York 1980)

2.34 D. Pieper: The Kinematics of Manipulators Under Computer Control, Ph.D. Thesis (Stanford University, Stanford 1968)

2.35 C.S.G. Lee: Robot arm kinematics, dynamics, and control, Computer **15**(12), 62–80 (1982)

2.36 M.T. Mason: *Mechanics of Robotic Manipulation* (MIT Press, Cambridge 2001)

2.37 H.Y. Lee, C.G. Liang: A new vector theory for the analysis of spatial mechanisms, Mech. Mach. Theory **23**(3), 209–217 (1988)

2.38 R. Manseur, K.L. Doty: A robot manipulator with 16 real inverse kinematic solutions, Int. J. Robotics Res. **8**(5), 75–79 (1989)

2.39 M. Raghavan, B. Roth: Kinematic analysis of the 6R manipulator of general geometry, 5th Int. Symp. Robotics Res. (1990)

2.40 D. Manocha, J. Canny: *Real Time Inverse Kinematics for General 6R Manipulators*, Tech. Rep. (University of California, Berkeley 1992)

2.41 B. Buchberger: Applications of Gröbner bases in non-linear computational geometry, Lect. Notes Comput. Sci. **296**, 52–80 (1989)

2.42 P. Kovacs: Minimum degree solutions for the inverse kinematics problem by application of the Buchberger algorithm. In: *Advances in Robot Kinematics*, ed. by S. Stifter, J. Lenarcic (Springer, New York 1991) pp. 326–334

2.43 L.W. Tsai, A.P. Morgan: Solving the kinematics of the most general six- and five-degree-of-freedom manipulators by continuation methods, ASME J. Mech. Transm. Autom. Des. **107**, 189–195 (1985)

2.44 C.W. Wampler, A.P. Morgan, A.J. Sommese: Numerical continuation methods for solving polynomial systems arising in kinematics, ASME J. Mech. Des. **112**, 59–68 (1990)

2.45 R. Manseur, K.L. Doty: Fast inverse kinematics of 5-revolute-axis robot manipulators, Mechan. Mach. Theory **27**(5), 587–597 (1992)

2.46 S.C.A. Thomopoulos, R.Y.J. Tam: An iterative solution to the inverse kinematics of robotic manipulators, Mechan. Mach. Theory **26**(4), 359–373 (1991)

2.47 J.J. Uicker Jr., J. Denavit, R.S. Hartenberg: An interactive method for the displacement analysis of spatial mechanisms, J. Appl. Mech. **31**, 309–314 (1964)

2.48 J. Zhao, N. Badler: Inverse kinematics positioning using nonlinear programming for highly articulated figures, Trans. Comput. Graph. **13**(4), 313–336 (1994)

2.49 D.E. Whitney: Resolved motion rate control of manipulators and human prostheses, IEEE Trans. Man Mach. Syst. **10**, 47–63 (1969)

2.50 H. Cheng, K. Gupta: A study of robot inverse kinematics based upon the solution of differential equations, J. Robotic Syst. **8**(2), 115–175 (1991)

2.51 L. Sciavicco, B. Siciliano: *Modeling and Control of Robot Manipulators* (Springer, London 2000)

2.52 R.S. Rao, A. Asaithambi, S.K. Agrawal: Inverse kinematic solution of robot manipulators using interval analysis, ASME J. Mech. Des. **120**(1), 147–150 (1998)
2.53 C.W. Wampler: Manipulator inverse kinematic solutions based on vector formulations and damped least squares methods, IEEE Trans. Syst. Man Cybern. **16**, 93–101 (1986)
2.54 D.E. Orin, W.W. Schrader: Efficient computation of the jacobian for robot manipulators, Int. J. Robotics Res. **3**(4), 66–75 (1984)
2.55 D.E. Whitney: The mathematics of coordinated control of prosthetic arms and manipulators, J. Dynamic Sys. Meas. Control **122**, 303–309 (1972)
2.56 R.P. Paul, B.E. Shimano, G. Mayer: Kinematic control equations for simple manipulators, IEEE Trans. Syst. Man Cybern. **11**(6), 339–455 (1981)
2.57 R.P. Paul, C.N. Stephenson: Kinematics of robot wrists, Int. J. Robotics Res. **20**(1), 31–38 (1983)
2.58 R.P. Paul, H. Zhang: Computationally efficient kinematics for manipulators with spherical wrists based on the homogeneous transformation representation, Int. J. Robotics Res. **5**(2), 32–44 (1986)
2.59 K.J. Waldron, K.H. Hunt: Series-parallel dualities in actively coordinated mechanisms, Int. J. Robotics Res. **10**, 473–480 (1991)
2.60 H. Asada, J.J.E. Slotine: *Robot Analysis and Control* (Wiley, New York 1986)
2.61 F.L. Lewis, C.T. Abdallah, D.M. Dawson: *Control of Robot Manipulators* (Macmillan, New York 1993)
2.62 R.J. Schilling: *Fundamentals of Robotics: Analysis and Control* (Prentice Hall, Englewood Cliffs 1990)
2.63 M.W. Spong, M. Vidyasagar: *Robot Dynamics and Control* (Wiley, New York 1989)
2.64 T. Yoshikawa: *Foundations of Robotics* (MIT Press, Cambridge 1990)

Avidbots Ex1016
Page 83 of 2154

Multimedia Contents



3. Dynamics

Roy Featherstone, David E. Orin

The dynamic equations of motion provide the relationships between actuation and contact forces acting on robot mechanisms, and the acceleration and motion trajectories that result. Dynamics is important for mechanical design, control, and simulation. A number of algorithms are important in these applications, and include computation of the following: *inverse dynamics*, *forward dynamics*, the *joint-space inertia matrix*, and the *operational-space inertia matrix*. This chapter provides efficient algorithms to perform each of these calculations on a rigid-body model of a robot mechanism. The algorithms are presented in their most general form and are applicable to robot mechanisms with general connectivity, geometry, and joint types. Such mechanisms include fixed-base robots, mobile robots, and parallel robot mechanisms.

In addition to the need for computational efficiency, algorithms should be formulated with a compact set of equations for ease of development and implementation. The use of spatial notation has been very effective in this regard, and is used in presenting the dynamics algorithms. Spatial vector algebra is a concise vector notation for describing rigid-body velocity, acceleration, inertia, etc., using six-dimensional (6-D) vectors and tensors.

The goal of this chapter is to introduce the reader to the subject of robot dynamics and to provide the reader with a rich set of algorithms, in a compact form, that they may apply to their particular robot mechanism. These algorithms are presented in tables for ready access.

3.1 **Overview** 38
3.1.1 Spatial Vector Notation 38
3.1.2 Canonical Equations 38
3.1.3 Dynamic Models of Rigid-Body Systems 38
3.1.4 Kinematic Trees 39
3.1.5 Kinematic Loops 39

3.2 **Spatial Vector Notation** 39
3.2.1 Motion and Force 40
3.2.2 Basis Vectors 40
3.2.3 Spatial Velocity and Force 40
3.2.4 Addition and Scalar Multiplication 41
3.2.5 Scalar Product 41
3.2.6 Coordinate Transforms 41
3.2.7 Vector Products 41
3.2.8 Differentiation 42
3.2.9 Acceleration 42
3.2.10 Spatial Momentum 42
3.2.11 Spatial Inertia 42
3.2.12 Equation of Motion 43
3.2.13 Computer Implementation 44
3.2.14 Summary 45

3.3 **Canonical Equations** 45
3.3.1 Joint-Space Formulation 45
3.3.2 Lagrange Formulation 46
3.3.3 Operational-Space Formulation 46
3.3.4 Impact Model 47

3.4 **Dynamic Models of Rigid-Body Systems** 47
3.4.1 Connectivity 47
3.4.2 Link Geometry 49
3.4.3 Link Inertias 49
3.4.4 Joint Models 49
3.4.5 Example System 50

3.5 **Kinematic Trees** 51
3.5.1 The Recursive Newton–Euler Algorithm 51
3.5.2 The Articulated-Body Algorithm 54
3.5.3 The Composite-Rigid-Body Algorithm 56
3.5.4 Operational-Space Inertia Matrix 57

3.6 **Kinematic Loops** 58
3.6.1 Formulation of Closed-Loop Algorithm 58
3.6.2 Closed-Loop Algorithm 60

3.7 **Conclusions and Further Reading** 61
3.7.1 Multibody Dynamics 62
3.7.2 Alternative Representations 62
3.7.3 Alternative Formulations 62
3.7.4 Efficiency 62
3.7.5 Accuracy 62
3.7.6 Software Packages 63
3.7.7 Symbolic Simplification 63
3.7.8 Algorithms for Parallel Computers .. 63
3.7.9 Topologically-Varying Systems 63

References 63

3.1 Overview

Robot dynamics provides the relationships between actuation and contact forces, and the acceleration and motion trajectories that result. The dynamic equations of motion provide the basis for a number of computational algorithms that are useful in mechanical design, control, and simulation. A growing area of their use is in computer animation of mobile systems, especially using human and humanoid models. In this Chapter, the fundamental dynamic relationships for robot mechanisms are presented, along with efficient algorithms for the most common computations. Spatial vector notation, a concise representation which makes use of 6-D vectors and tensors, is used in the algorithms.

This chapter presents efficient low-order algorithms for four major computations:

1. Inverse dynamics, in which the required joint actuator torques/forces are computed from a specification of the robot's trajectory (position, velocity, and acceleration),
2. Forward dynamics in which the applied joint actuator torques/forces are specified and the joint accelerations are to be determined,
3. The joint-space inertia matrix, which maps the joint accelerations to the joint torques/forces, and
4. The operational-space inertia matrix, which maps task accelerations to task forces in operational or Cartesian space.

Inverse dynamics is used in feedforward control and trajectory planning. Forward dynamics is required for simulation. The joint-space inertia (mass) matrix is used in analysis, in feedback control to linearize the dynamics, and is an integral part of many forward dynamics formulations. The operational-space inertia matrix is used in control at the task or end-effector level.

3.1.1 Spatial Vector Notation

Section 3.2 presents the spatial vector notation, which is used to express the algorithms in this chapter in a clear and concise manner. It was originally developed by *Featherstone* [3.1] to provide a concise vector notation for describing rigid-body velocity, acceleration, inertia, etc., using 6-D vectors and tensors. Section 3.2 explains the meanings of spatial vectors and operators, and provides a detailed tabulation of the correspondence between spatial and standard three-dimensional (3-D) quantities and operators, so that the algorithms in the later sections can be understood. Formulae for efficient computer implementation of spatial arithmetic are also provided. Effort is taken in the discussion of spatial vectors to distinguish between the coordinate vectors and the quantities they represent. This illuminates some of the important characteristics of spatial vectors.

3.1.2 Canonical Equations

The dynamic equations of motion are provided in Sect. 3.3 in two fundamental forms: the joint-space formulation and the operational-space formulation. The terms in the joint-space formulation have traditionally been derived using a Lagrangian approach in which they are developed independently of any reference coordinate frame. The Lagrange formulation provides a description of the relationship between the joint actuator forces and the motion of the mechanism, and fundamentally operates on the kinetic and potential energy in the system. The resulting joint-space formulation has a number of notable properties that have proven useful for developing control algorithms. The equations to relate the terms in the joint-space and operational-space formulations, along with an impact model, are also provided in this section.

3.1.3 Dynamic Models of Rigid-Body Systems

The algorithms in this chapter are model-based and require a data structure describing a robot mechanism as one of their input arguments. Section 3.4 gives a description of the components of this model: a connectivity graph, link geometry parameters, link inertia parameters, and a set of joint models. The description of the connectivity is general so that it covers both kinematic trees and closed-loop mechanisms. Kinematic trees and the spanning tree for a closed-loop mechanism share a common notation. In order to describe

the link and joint geometry, two coordinate frames are associated with each joint, one each attached to the *predecessor* and *successor* links. These frames are defined to be compatible with the modified Denavit–Hartenberg convention of *Craig* [3.2] for serial mechanisms containing single-degree-of-freedom (DOF) joints, but are nevertheless applicable to general rigid-body systems containing general multi-DOF joints. The relationship between connected links is described using the general joint model of *Roberson* and *Schwertassek* [3.3]. A humanoid robot is given as an example to illustrate the link and joint numbering scheme, as well as the assignment of coordinate frames to describe the links and joints. The example includes a floating base, and revolute, universal, and spherical joints.

3.1.4 Kinematic Trees

The algorithms presented in Sect. 3.5 calculate the inverse dynamics, forward dynamics, joint-space inertia matrix, and operational-space inertia matrix for any robot mechanism that is a kinematic tree. An $O(n)$ algorithm for inverse dynamics is provided, where n is the number of degrees of freedom in the mechanism. It uses a Newton–Euler formulation of the problem, and is based on the very efficient recursive Newton–Euler algorithm (RNEA) of *Luh* et al. [3.4]. Two algorithms are provided for forward dynamics. The first is the $O(n)$ articulated-body algorithm (ABA) which was developed by *Featherstone* [3.1]. The second is the $O(n^2)$ composite-rigid-body algorithm (CRBA), developed by *Walker* and *Orin* [3.5], to compute the joint-space inertia matrix (JSIM). This matrix, together with a vector computed using the RNEA, provide the coefficients of the equation of motion, which can then be solved directly for the accelerations [3.5]. The operational-space inertia matrix (OSIM) is a kind of articulated-body inertia, and two algorithms are given to calculate it. The first uses the basic definition of the OSIM, and the second is a straightforward $O(n)$ algorithm which is based on efficient solution of the forward dynamics problem. The inputs, outputs, model data, and pseudocode for each algorithm are summarized in tables for ready access.

3.1.5 Kinematic Loops

The above algorithms apply only to mechanisms having the connectivity of kinematic trees, including unbranched kinematic chains. A final algorithm is provided in Sect. 3.6 for the forward dynamics of closed-loop systems, including parallel robot mechanisms. The algorithm makes use of the dynamic equations of motion for a spanning tree of the closed-loop system, and supplements these with loop-closure constraint equations. Three different methods are outlined to solve the resulting linear system of equations. Method 2 is particularly useful if $n \gg n^c$, where n^c is the number of constraints due to the loop-closing joints. This method offers the opportunity to use $O(n)$ algorithms on the spanning tree [3.6]. The section ends with an efficient algorithm to compute the loop-closure constraints by transforming them to a single coordinate system. Since the loop-closure constraint equations are applied at the acceleration level, standard *Baumgarte* stabilization [3.7] is used to prevent the accumulation of position and velocity errors in the loop-closure constraints.

The final section in this chapter provides a conclusion and suggestions for further reading. The area of robot dynamics has been, and continues to be, a very rich area of investigation. This section outlines the major contributions that have been made in the area and the work most often cited. Unfortunately, space does not permit us to provide a comprehensive review of the extensive literature in the area.

3.2 Spatial Vector Notation

There is no single standard notation for robot dynamics. The notations currently in use include 3-D vectors, 4×4 matrices, and several types of 6-D vector: screws, motors, Lie algebra elements, and spatial vectors. Six-dimensional vector notations are generally the best, being more compact than 3-D vectors, and more powerful than 4×4 matrices. We therefore use 6-D vectors throughout this chapter. In particular, we shall use the spatial vector algebra described in [3.8]. This section provides a brief summary of spatial vectors. Descriptions of 4×4 matrix notations can be found in [3.2, 9], and descriptions of other 6-D vector notations can be found in [3.10–12].

In this handbook, vectors are usually denoted by bold italic letters (e.g., $\boldsymbol{f}$, $\boldsymbol{v}$). However, to avoid a few name clashes, we shall use upright bold letters to denote spatial vectors (e.g., $\mathbf{f}$, $\mathbf{v}$). Note that this applies only to vectors, not tensors. Also, *in this section only*, we will underline coordinate vectors to distinguish them from the vectors that they represent (e.g., $\underline{\boldsymbol{v}}$ and $\underline{\mathbf{v}}$, representing $\boldsymbol{v}$ and $\mathbf{v}$).

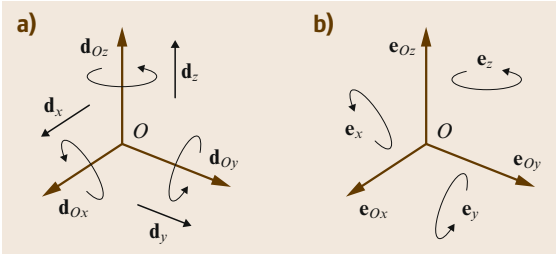


Fig.3.1a,b Plücker basis vectors for motions **(a)** and forces **(b)**

3.2.1 Motion and Force

For mathematical reasons, it is useful to distinguish between those vectors that describe the motions of rigid bodies, and those that describe the forces acting upon them. We therefore place motion vectors in a vector space called M^6, and force vectors in a space called F^6. (The superscripts indicate the dimension.) Motion vectors describe quantities like velocity, acceleration, infinitesimal displacement, and directions of motion freedom; force vectors describe force, momentum, contact normals, and so on.

3.2.2 Basis Vectors

Suppose that $\boldsymbol{v}$ is a 3-D vector, and that $\underline{\boldsymbol{v}} = (v_x, v_y, v_z)^{\mathsf{T}}$ is the Cartesian coordinate vector that represents $\boldsymbol{v}$ in the orthonormal basis $\{\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}, \hat{\boldsymbol{z}}\}$. The relationship between $\boldsymbol{v}$ and $\underline{\boldsymbol{v}}$ is then given by the formula

$$\boldsymbol{v} = \hat{\boldsymbol{x}} v_x + \hat{\boldsymbol{y}} v_y + \hat{\boldsymbol{z}} v_z \,.$$

This same idea applies also to spatial vectors, except that we use Plücker coordinates instead of Cartesian coordinates, and a Plücker basis instead of an orthonormal basis.

Plücker coordinates were introduced in Sect. 2.2.6, but the basis vectors are shown in Fig. 3.1. There are 12 basis vectors in total: six for motion vectors and six for forces. Given a Cartesian coordinate frame, O_{xyz}, the Plücker basis vectors are defined as follows: three unit rotations about the directed lines Ox, Oy, and Oz, denoted by $\mathbf{d}_{Ox}$, $\mathbf{d}_{Oy}$, and $\mathbf{d}_{Oz}$, three unit translations in the directions x, y, and z, denoted by $\mathbf{d}_x$, $\mathbf{d}_y$, and $\mathbf{d}_z$, three unit couples about the x, y, and z directions, denoted by $\mathbf{e}_x$, $\mathbf{e}_y$, and $\mathbf{e}_z$, and three unit forces along the lines Ox, Oy, and Oz, denoted by $\mathbf{e}_{Ox}$, $\mathbf{e}_{Oy}$, and $\mathbf{e}_{Oz}$.

3.2.3 Spatial Velocity and Force

Given any point O, the velocity of a rigid body can be described by a pair of 3-D vectors, $\boldsymbol{\omega}$ and $\boldsymbol{v}_O$, which specify the body's angular velocity and the linear velocity of the body-fixed point currently at O. Note that $\boldsymbol{v}_O$ is not the velocity of O itself, but the velocity of the body-fixed point that happens to coincide with O at the current instant.

The velocity of this same rigid body can also be described by a single spatial motion vector, $\mathbf{v} \in \mathsf{M}^6$. To obtain $\mathbf{v}$ from $\boldsymbol{\omega}$ and $\boldsymbol{v}_O$, we first introduce a Cartesian frame, O_{xyz}, with its origin at O. This frame defines a Cartesian coordinate system for $\boldsymbol{\omega}$ and $\boldsymbol{v}_O$, and also a Plücker coordinate system for $\mathbf{v}$. Given these coordinate systems, it can be shown that

$$\begin{aligned}\mathbf{v} = {} & \mathbf{d}_{Ox}\omega_x + \mathbf{d}_{Oy}\omega_y + \mathbf{d}_{Oz}\omega_z + \mathbf{d}_x v_{Ox} \\ & + \mathbf{d}_y v_{Oy} + \mathbf{d}_z v_{Oz} \,, \end{aligned} \tag{3.1}$$

where $\omega_x, \ldots, v_{Oz}$ are the Cartesian coordinates of $\boldsymbol{\omega}$ and $\boldsymbol{v}_O$ in O_{xyz}. Thus, the Plücker coordinates of $\mathbf{v}$ are the Cartesian coordinates of $\boldsymbol{\omega}$ and $\boldsymbol{v}_O$. The coordinate vector representing $\mathbf{v}$ in O_{xyz} can be written

$$\underline{\mathbf{v}}_O = \begin{pmatrix} \omega_x \\ \vdots \\ v_{Oz} \end{pmatrix} = \begin{pmatrix} \underline{\boldsymbol{\omega}} \\ \underline{\boldsymbol{v}}_O \end{pmatrix} . \tag{3.2}$$

The notation on the far right of this equation is simply a convenient abbreviation of the list of Plücker coordinates.

The definition of spatial force is very similar. Given any point O, any system of forces acting on a single rigid body is equivalent to a single force $\boldsymbol{f}$ acting on a line passing through O, together with a pure couple, $\boldsymbol{n}_O$, which is the moment of the force system about O. Thus, the two vectors $\boldsymbol{f}$ and $\boldsymbol{n}_O$ describe the force acting on a rigid body in much the same way that $\boldsymbol{\omega}$ and $\boldsymbol{v}_O$ describe its velocity. This same force can also be described by a single spatial force vector, $\mathbf{f} \in \mathsf{F}^6$. Introducing the frame O_{xyz}, as before, it can be shown that

$$\begin{aligned}\mathbf{f} = {} & \mathbf{e}_x n_{Ox} + \mathbf{e}_y n_{Oy} + \mathbf{e}_z n_{Oz} + \mathbf{e}_{Ox} f_x \\ & + \mathbf{e}_{Oy} f_y + \mathbf{e}_{Oz} f_z \,, \end{aligned} \tag{3.3}$$

where $n_{Ox}, \ldots, f_z$ are the Cartesian coordinates of $\boldsymbol{n}_O$ and $\boldsymbol{f}$ in O_{xyz}. The coordinate vector representing $\mathbf{f}$ in O_{xyz} can then be written

$$\underline{\mathbf{f}}_O = \begin{pmatrix} n_{Ox} \\ \vdots \\ f_z \end{pmatrix} = \begin{pmatrix} \underline{\boldsymbol{n}}_O \\ \underline{\boldsymbol{f}} \end{pmatrix} . \tag{3.4}$$

Again, these are the Plücker coordinates of $\mathbf{f}$ in O_{xyz}, and the notation on the far right is simply a convenient abbreviation of the list of Plücker coordinates.

3.2.4 Addition and Scalar Multiplication

Spatial vectors behave in the obvious way under addition and scalar multiplication. For example, if $\mathbf{f}_1$ and $\mathbf{f}_2$ both act on the same rigid body, then their resultant is $\mathbf{f}_1 + \mathbf{f}_2$; if two different bodies have velocities of $\mathbf{v}_1$ and $\mathbf{v}_2$, then the velocity of the second body relative to the first is $\mathbf{v}_2 - \mathbf{v}_1$; and if $\mathbf{f}$ denotes a force of 1 N acting along a particular line in space, then $\alpha\,\mathbf{f}$ denotes a force of α N acting along the same line.

3.2.5 Scalar Product

A scalar product is defined between any two spatial vectors, provided that one of them is a motion and the other a force. Given any $\mathbf{m} \in \mathsf{M}^6$ and $\mathbf{f} \in \mathsf{F}^6$, the scalar product can be written either $\mathbf{f}\cdot\mathbf{m}$ or $\mathbf{m}\cdot\mathbf{f}$, and expresses the work done by a force $\mathbf{f}$ acting on a body with motion $\mathbf{m}$. Expressions like $\mathbf{f}\cdot\mathbf{f}$ and $\mathbf{m}\cdot\mathbf{m}$ are not defined. If $\underline{\mathbf{m}}$ and $\underline{\mathbf{f}}$ are coordinate vectors representing $\mathbf{m}$ and $\mathbf{f}$ in the same coordinate system, then

$$\mathbf{m}\cdot\mathbf{f} = \underline{\mathbf{m}}^{\mathrm{T}}\underline{\mathbf{f}}\,. \tag{3.5}$$

3.2.6 Coordinate Transforms

Motion and force vectors obey different transformation rules. Let A and B be two coordinate frames, each defining a coordinate system of the same name; and let $\underline{\mathbf{m}}_A$, $\underline{\mathbf{m}}_B$, $\underline{\mathbf{f}}_A$, and $\underline{\mathbf{f}}_B$ be coordinate vectors representing the spatial vectors $\mathbf{m} \in \mathsf{M}^6$ and $\mathbf{f} \in \mathsf{F}^6$ in A and B coordinates, respectively. The transformation rules are then

$$\underline{\mathbf{m}}_B = {}^B\boldsymbol{X}_A\,\underline{\mathbf{m}}_A \tag{3.6}$$

and

$$\underline{\mathbf{f}}_B = {}^B\boldsymbol{X}_A^F\,\underline{\mathbf{f}}_A\,, \tag{3.7}$$

where ${}^B\boldsymbol{X}_A$ and ${}^B\boldsymbol{X}_A^F$ are the coordinate transformation matrices from A to B for motion and force vectors, respectively. These matrices are related by the identity

$${}^B\boldsymbol{X}_A^F \equiv ({}^B\boldsymbol{X}_A)^{-T} \equiv ({}^A\boldsymbol{X}_B)^{\mathrm{T}}\,. \tag{3.8}$$

Suppose that the position and orientation of frame A relative to frame B is described by a position vector ${}^B\underline{\boldsymbol{p}}_A$ and a 3×3 rotation matrix ${}^B\boldsymbol{R}_A$, as described in Sect. 2.2. The formula for ${}^B\boldsymbol{X}_A$ is then

$$\begin{aligned} {}^B\boldsymbol{X}_A &= \begin{pmatrix} \mathbf{1} & \mathbf{0} \\ \boldsymbol{S}({}^B\underline{\boldsymbol{p}}_A) & \mathbf{1} \end{pmatrix} \begin{pmatrix} {}^B\boldsymbol{R}_A & \mathbf{0} \\ \mathbf{0} & {}^B\boldsymbol{R}_A \end{pmatrix} \\ &= \begin{pmatrix} {}^B\boldsymbol{R}_A & \mathbf{0} \\ \boldsymbol{S}({}^B\underline{\boldsymbol{p}}_A){}^B\boldsymbol{R}_A & {}^B\boldsymbol{R}_A \end{pmatrix}, \end{aligned} \tag{3.9}$$

and its inverse is

$${}^A\boldsymbol{X}_B = \begin{pmatrix} {}^A\boldsymbol{R}_B & \mathbf{0} \\ \mathbf{0} & {}^A\boldsymbol{R}_B \end{pmatrix} \begin{pmatrix} \mathbf{1} & \mathbf{0} \\ -\boldsymbol{S}({}^B\underline{\boldsymbol{p}}_A) & \mathbf{1} \end{pmatrix}. \tag{3.10}$$

The quantity $\boldsymbol{S}(\underline{\boldsymbol{p}})$ is the skew-symmetric matrix that satisfies $\boldsymbol{S}(\underline{\boldsymbol{p}})\underline{\boldsymbol{v}} = \underline{\boldsymbol{p}} \times \underline{\boldsymbol{v}}$ for any 3-D vector $\underline{\boldsymbol{v}}$. It is defined by the equation

$$\boldsymbol{S}(\underline{\boldsymbol{p}}) = \begin{pmatrix} 0 & -p_z & p_y \\ p_z & 0 & -p_x \\ -p_y & p_x & 0 \end{pmatrix}. \tag{3.11}$$

3.2.7 Vector Products

There are two vector (cross) products defined on spatial vectors. The first takes two motion-vector arguments, and produces a motion-vector result. It is defined by the formula

$$\begin{aligned} \underline{\mathbf{m}}_1 \times \underline{\mathbf{m}}_2 &= \begin{pmatrix} \underline{\boldsymbol{m}}_1 \\ \underline{\boldsymbol{m}}_{1O} \end{pmatrix} \times \begin{pmatrix} \underline{\boldsymbol{m}}_2 \\ \underline{\boldsymbol{m}}_{2O} \end{pmatrix} \\ &= \begin{pmatrix} \underline{\boldsymbol{m}}_1 \times \underline{\boldsymbol{m}}_2 \\ \underline{\boldsymbol{m}}_1 \times \underline{\boldsymbol{m}}_{2O} + \underline{\boldsymbol{m}}_{1O} \times \underline{\boldsymbol{m}}_2 \end{pmatrix}. \end{aligned} \tag{3.12}$$

The second takes a motion vector as left-hand argument and a force vector as right-hand argument, and produces a force-vector result. It is defined by the formula

$$\begin{aligned} \underline{\mathbf{m}} \times \underline{\mathbf{f}} &= \begin{pmatrix} \underline{\boldsymbol{m}} \\ \underline{\boldsymbol{m}}_O \end{pmatrix} \times \begin{pmatrix} \underline{\boldsymbol{f}}_O \\ \underline{\boldsymbol{f}} \end{pmatrix} \\ &= \begin{pmatrix} \underline{\boldsymbol{m}} \times \underline{\boldsymbol{f}}_O + \underline{\boldsymbol{m}}_O \times \underline{\boldsymbol{f}} \\ \underline{\boldsymbol{m}} \times \underline{\boldsymbol{f}} \end{pmatrix}. \end{aligned} \tag{3.13}$$

These products arise in differentiation formulae.

It is possible to define a spatial cross-product operator, in analogy with (3.11), as follows

$$\boldsymbol{S}(\underline{\mathbf{m}}) = \begin{pmatrix} \boldsymbol{S}(\underline{\boldsymbol{m}}) & \mathbf{0} \\ \boldsymbol{S}(\underline{\boldsymbol{m}}_O) & \boldsymbol{S}(\underline{\boldsymbol{m}}) \end{pmatrix}, \tag{3.14}$$

in which case

$$\underline{\mathbf{m}}_1 \times \underline{\mathbf{m}}_2 = \boldsymbol{S}(\underline{\mathbf{m}}_1)\,\underline{\mathbf{m}}_2\,, \tag{3.15}$$

but

$$\underline{\mathbf{m}} \times \underline{\mathbf{f}} = -\boldsymbol{S}(\underline{\mathbf{m}})^{\mathrm{T}}\underline{\mathbf{f}}\,. \tag{3.16}$$

Observe that $\boldsymbol{S}(\underline{\mathbf{m}})$ maps motion vectors to motion vectors, but $\boldsymbol{S}(\underline{\boldsymbol{m}})^{\mathrm{T}}$ maps force vectors to force vectors.

3.2.8 Differentiation

The derivative of a spatial vector is defined by

$$\frac{\mathrm{d}}{\mathrm{d}x}\mathbf{s}(x) = \lim_{\delta x \to 0} \frac{\mathbf{s}(x+\delta x) - \mathbf{s}(x)}{\delta x}\,, \tag{3.17}$$

where $\mathbf{s}$ here stands for any spatial vector. The derivative is a spatial vector of the same kind (motion or force) as that being differentiated.

The formula for differentiating a spatial coordinate vector in a moving coordinate system is

$$\left(\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{s}\right)_A = \frac{\mathrm{d}}{\mathrm{d}t}\underline{\mathbf{s}}_A + \underline{\mathbf{v}}_A \times \underline{\mathbf{s}}_A\,, \tag{3.18}$$

where $\mathbf{s}$ is any spatial vector, $\mathrm{d}\mathbf{s}/\mathrm{d}t$ is the time derivative of $\mathbf{s}$, A is the moving coordinate system, $(\mathrm{d}\mathbf{s}/\mathrm{d}t)_A$ is the coordinate vector that represents $\mathrm{d}\mathbf{s}/\mathrm{d}t$ in A coordinates, $\underline{\mathbf{s}}_A$ is the coordinate vector that represents $\mathbf{s}$ in A coordinates, $\mathrm{d}\underline{\mathbf{s}}_A/\mathrm{d}t$ is the time derivative of $\underline{\mathbf{s}}_A$ (which is the componentwise derivative, since $\underline{\mathbf{s}}_A$ is a coordinate vector), and $\underline{\mathbf{v}}_A$ is the velocity of the A coordinate frame, expressed in A coordinates.

The time derivative of a spatial vector that changes only because it is moving is given by

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{s} = \mathbf{v} \times \mathbf{s}\,, \tag{3.19}$$

where $\mathbf{v}$ is the velocity of $\mathbf{s}$. This formula is useful for differentiating quantities that do not change in their own right, but are attached to moving rigid bodies (e.g., joint axis vectors).

3.2.9 Acceleration

Spatial acceleration is defined as the rate of change of spatial velocity. Unfortunately, this means that spatial acceleration differs from the classical textbook definition of rigid-body acceleration, which we shall call *classical acceleration*. Essentially, the difference can be summarized as follows

$$\underline{\mathbf{a}} = \begin{pmatrix} \dot{\underline{\boldsymbol{\omega}}} \\ \dot{\underline{\boldsymbol{v}}}_O \end{pmatrix} \quad \text{and} \quad \underline{\mathbf{a}}' = \begin{pmatrix} \dot{\underline{\boldsymbol{\omega}}} \\ \dot{\underline{\boldsymbol{v}}}'_O \end{pmatrix}, \tag{3.20}$$

where $\underline{\mathbf{a}}$ is the spatial acceleration, $\underline{\mathbf{a}}'$ is the classical acceleration, $\dot{\underline{\boldsymbol{v}}}_O$ is the derivative of $\underline{\boldsymbol{v}}_O$ taking O to be fixed in space, and $\dot{\underline{\boldsymbol{v}}}'_O$ is the derivative of $\underline{\boldsymbol{v}}_O$ taking O to be fixed in the body. The two accelerations are related by

$$\underline{\mathbf{a}}' = \underline{\mathbf{a}} + \begin{pmatrix} \mathbf{0} \\ \underline{\boldsymbol{\omega}} \times \underline{\boldsymbol{v}}_O \end{pmatrix}. \tag{3.21}$$

If $\boldsymbol{r}$ is a position vector giving the position of the body-fixed point at O relative to any fixed point, then

$$\begin{aligned} \boldsymbol{v}_O &= \dot{\boldsymbol{r}}\,, \\ \dot{\boldsymbol{v}}'_O &= \ddot{\boldsymbol{r}}\,, \\ \dot{\boldsymbol{v}}_O &= \ddot{\boldsymbol{r}} - \boldsymbol{\omega} \times \boldsymbol{v}_O\,. \end{aligned} \tag{3.22}$$

The practical difference is that spatial accelerations are easier to use. For example, if the bodies B_1 and B_2 have velocities of $\mathbf{v}_1$ and $\mathbf{v}_2$, respectively, and $\mathbf{v}_{\mathrm{rel}}$ is the relative velocity of B_2 with respect to B_1, then

$$\mathbf{v}_2 = \mathbf{v}_1 + \mathbf{v}_{\mathrm{rel}}\,.$$

The relationship between their spatial accelerations is obtained simply by differentiating the velocity formula

$$\frac{\mathrm{d}}{\mathrm{d}t}(\mathbf{v}_2 = \mathbf{v}_1 + \mathbf{v}_{\mathrm{rel}}) \quad \Rightarrow \quad \mathbf{a}_2 = \mathbf{a}_1 + \mathbf{a}_{\mathrm{rel}}\,.$$

Observe that spatial accelerations are composed by addition, exactly like velocities. There are no Coriolis or centrifugal terms to worry about. This is a significant improvement on the formulae for composing classical accelerations, such as those in [3.2, 13, 14].

3.2.10 Spatial Momentum

Suppose that a rigid body has a mass of m, a center of mass at C, and a rotational inertia of $\bar{\boldsymbol{I}}^{\mathrm{cm}}$ about C (Fig. 3.2). If this body is moving with a spatial velocity of $\underline{\mathbf{v}}_C = (\underline{\boldsymbol{\omega}}^{\mathrm{T}}\underline{\boldsymbol{v}}_C^{\mathrm{T}})^{\mathrm{T}}$, then its linear momentum is $\boldsymbol{h} = m\boldsymbol{v}_C$, and its intrinsic angular momentum is $\boldsymbol{h}_C = \bar{\boldsymbol{I}}^{\mathrm{cm}}\boldsymbol{\omega}$. Its moment of momentum about a general point, O, is $\boldsymbol{h}_O = \boldsymbol{h}_C + \boldsymbol{c} \times \boldsymbol{h}$, where $\boldsymbol{c} = \overrightarrow{OC}$. We can assemble these vectors into a spatial momentum vector as follows

$$\underline{\mathbf{h}}_C = \begin{pmatrix} \underline{\boldsymbol{h}}_C \\ \underline{\boldsymbol{h}} \end{pmatrix} = \begin{pmatrix} \bar{\underline{\boldsymbol{I}}}^{\mathrm{cm}}\underline{\boldsymbol{\omega}} \\ m\underline{\boldsymbol{v}}_C \end{pmatrix} \tag{3.23}$$

and

$$\underline{\mathbf{h}}_O = \begin{pmatrix} \underline{\boldsymbol{h}}_O \\ \underline{\boldsymbol{h}} \end{pmatrix} = \begin{pmatrix} \mathbf{1} & S(\underline{\boldsymbol{c}}) \\ \mathbf{0} & \mathbf{1} \end{pmatrix} \underline{\mathbf{h}}_C\,. \tag{3.24}$$

Spatial momentum is a force vector, and transforms accordingly.

3.2.11 Spatial Inertia

The spatial momentum of a rigid body is the product of its spatial inertia and velocity

$$\mathbf{h} = \boldsymbol{I}\mathbf{v}\,, \tag{3.25}$$

where $\boldsymbol{I}$ is the spatial inertia. Expressed in Plücker coordinates at C, we have

$$\mathbf{h}_C = \boldsymbol{I}_C \mathbf{v}_C \,, \tag{3.26}$$

which implies

$$\boldsymbol{I}_C = \begin{pmatrix} \bar{\boldsymbol{I}}^{\mathrm{cm}} & \mathbf{0} \\ \mathbf{0} & m\mathbf{1} \end{pmatrix} . \tag{3.27}$$

This is the general formula for the spatial inertia of a rigid body expressed at its center of mass. To express it at another point, O, we proceed as follows. From (3.24), (3.26), and (3.27)

$$\begin{aligned} \underline{\mathbf{h}}_O &= \begin{pmatrix} \mathbf{1} & \boldsymbol{S}(\underline{\boldsymbol{c}}) \\ \mathbf{0} & \mathbf{1} \end{pmatrix} \begin{pmatrix} \bar{\boldsymbol{I}}^{\mathrm{cm}} & \mathbf{0} \\ \mathbf{0} & m\mathbf{1} \end{pmatrix} \underline{\mathbf{v}}_C \\ &= \begin{pmatrix} \mathbf{1} & \boldsymbol{S}(\underline{\boldsymbol{c}}) \\ \mathbf{0} & \mathbf{1} \end{pmatrix} \begin{pmatrix} \bar{\boldsymbol{I}}^{\mathrm{cm}} & \mathbf{0} \\ \mathbf{0} & m\mathbf{1} \end{pmatrix} \begin{pmatrix} \mathbf{1} & \mathbf{0} \\ \boldsymbol{S}(\underline{\boldsymbol{c}})^{\mathrm{T}} & \mathbf{1} \end{pmatrix} \underline{\mathbf{v}}_O \\ &= \begin{pmatrix} \bar{\boldsymbol{I}}^{\mathrm{cm}} + m\boldsymbol{S}(\underline{\boldsymbol{c}})\boldsymbol{S}(\underline{\boldsymbol{c}})^{\mathrm{T}} & m\boldsymbol{S}(\underline{\boldsymbol{c}}) \\ m\boldsymbol{S}(\underline{\boldsymbol{c}})^{\mathrm{T}} & m\mathbf{1} \end{pmatrix} \underline{\mathbf{v}}_O \,; \end{aligned}$$

but we also have that $\underline{\mathbf{h}}_O = \boldsymbol{I}_O \underline{\mathbf{v}}_O$, so

$$\boldsymbol{I}_O = \begin{pmatrix} \bar{\boldsymbol{I}}^{\mathrm{cm}} + m\boldsymbol{S}(\underline{\boldsymbol{c}})\boldsymbol{S}(\underline{\boldsymbol{c}})^{\mathrm{T}} & m\boldsymbol{S}(\underline{\boldsymbol{c}}) \\ m\boldsymbol{S}(\underline{\boldsymbol{c}})^{\mathrm{T}} & m\mathbf{1} \end{pmatrix} . \tag{3.28}$$

This equation can also be written

$$\boldsymbol{I}_O = \begin{pmatrix} \bar{\boldsymbol{I}}_O & m\boldsymbol{S}(\underline{\boldsymbol{c}}) \\ m\boldsymbol{S}(\underline{\boldsymbol{c}})^{\mathrm{T}} & m\mathbf{1} \end{pmatrix} , \tag{3.29}$$

where

$$\bar{\boldsymbol{I}}_O = \bar{\boldsymbol{I}}^{\mathrm{cm}} + m\boldsymbol{S}(\underline{\boldsymbol{c}})\boldsymbol{S}(\underline{\boldsymbol{c}})^{\mathrm{T}} \tag{3.30}$$

is the rotational inertia of the rigid body about O.

Spatial inertia matrices are symmetric and positive-definite. In the general case, 21 numbers are required to specify a spatial inertia (e.g., for an articulated-body or operational-space inertia); but a rigid-body inertia needs only 10 parameters: the mass, the coordinates of the center of mass, and the six independent elements of either $\bar{\boldsymbol{I}}^{\mathrm{cm}}$ or $\bar{\boldsymbol{I}}_O$.

The transformation rule for spatial inertias is

$$\boldsymbol{I}_B = {}^B\boldsymbol{X}_A^F \, \boldsymbol{I}_A \, {}^A\boldsymbol{X}_B \,, \tag{3.31}$$

where A and B are any two coordinate systems. In practice, we often need to calculate $\boldsymbol{I}_A$ from $\boldsymbol{I}_B$, given only ${}^B\boldsymbol{X}_A$. The formula for this transformation is

$$\boldsymbol{I}_A = ({}^B\boldsymbol{X}_A)^{\mathrm{T}} \boldsymbol{I}_B \, {}^B\boldsymbol{X}_A \,. \tag{3.32}$$

If two bodies, having inertias $\boldsymbol{I}_1$ and $\boldsymbol{I}_2$, are rigidly connected to form a single composite body, then the inertia of the composite, $\boldsymbol{I}_{\mathrm{tot}}$, is the sum of the inertias of its parts

$$\boldsymbol{I}_{\mathrm{tot}} = \boldsymbol{I}_1 + \boldsymbol{I}_2 \,. \tag{3.33}$$

This single equation takes the place of three equations in the traditional 3-D vector approach: one to compute the composite mass, one to compute the composite center of mass, and one to compute the composite rotational inertia. If a rigid body with inertia $\boldsymbol{I}$ is moving with a velocity of $\mathbf{v}$, then its kinetic energy is

$$T = \frac{1}{2} \mathbf{v} \cdot \boldsymbol{I}\mathbf{v} \,. \tag{3.34}$$

If a rigid body, B, is part of a larger system, then it is possible to define an apparent-inertia matrix for B, which describes the relationship between a force acting on B and its resulting acceleration, taking into account the effects of the other bodies in the system. Such quantities are called articulated-body inertias. If B happens to be the end-effector of a robot, then its apparent inertia is called an operational-space inertia.

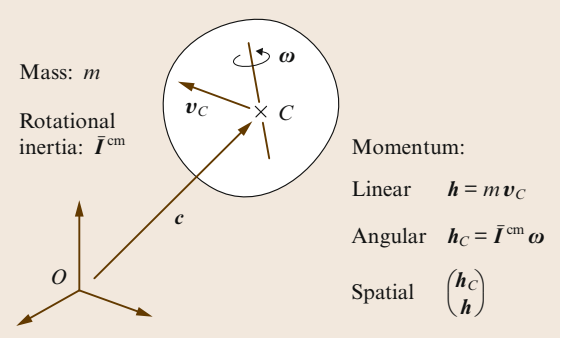


Fig. 3.2 Spatial momentum

3.2.12 Equation of Motion

The spatial equation of motion states that the net force acting on a rigid body equals its rate of change of momentum

$$\mathbf{f} = \frac{\mathrm{d}}{\mathrm{d}t}(\boldsymbol{I}\mathbf{v}) = \boldsymbol{I}\mathbf{a} + \dot{\boldsymbol{I}}\mathbf{v} \,.$$

It can be shown that the expression $\dot{\boldsymbol{I}}\mathbf{v}$ evaluates to $(\mathbf{v} \times \boldsymbol{I}\mathbf{v})$ [3.8, 15], so the equation of motion can be written

$$\mathbf{f} = \boldsymbol{I}\mathbf{a} + \mathbf{v} \times \boldsymbol{I}\mathbf{v} \,. \tag{3.35}$$

This single equation incorporates both Newton's and Euler's equations of motion for a rigid body. To

Table 3.1 Summary of spatial vector notation

Spatial quantities:	
$\mathbf{v}$	Velocity of a rigid body
$\mathbf{a}$	Spatial acceleration of a rigid body ($\mathbf{a} = \dot{\mathbf{v}}$)
$\mathbf{a}'$	Classical description of rigid-body acceleration expressed as a 6-D vector
$\mathbf{f}$	Force acting on a rigid body
$\boldsymbol{I}$	Inertia of a rigid body
$\boldsymbol{X}$	Plücker coordinate transform for motion vectors
$\boldsymbol{X}^F$	Plücker coordinate transform for force vectors ($\boldsymbol{X}^F = \boldsymbol{X}^{-T}$)
${}^B\boldsymbol{X}_A$	Plücker transform from A coordinates to B coordinates
$\mathbf{m}$	A generic motion vector (any element of M^6)
3-D quantities:	
O	Coordinate system origin
$\boldsymbol{r}$	Position of the body-fixed point at O relative to any fixed point in space
$\boldsymbol{\omega}$	Angular velocity of a rigid body
$\boldsymbol{v}_O$	Linear velocity of the body-fixed point at O ($\boldsymbol{v}_O = \dot{\boldsymbol{r}}$)
$\dot{\boldsymbol{\omega}}$	Angular acceleration of a rigid body
$\dot{\boldsymbol{v}}_O$	The derivative of $\boldsymbol{v}_O$ taking O to be fixed in space
$\dot{\boldsymbol{v}}'_O$	The derivative of $\boldsymbol{v}_O$ taking O to be fixed in the body; the classical acceleration of the body-fixed point at O ($\dot{\boldsymbol{v}}'_O = \ddot{\boldsymbol{r}}$)
$\boldsymbol{f}$	Linear force acting on a rigid body, or the resultant of a system of linear forces
$\boldsymbol{n}_O$	Moment about O of a linear force or system of linear forces
m	Mass of a rigid body
$\boldsymbol{c}$	Position of a rigid body's center of mass, measured relative to O
$\boldsymbol{h}$	First moment of mass of a rigid body, $\boldsymbol{h} = m\boldsymbol{c}$; can also denote linear momentum
$\bar{\boldsymbol{I}}^{\mathrm{cm}}$	Moment of inertia about a body's centre of mass
$\bar{\boldsymbol{I}}$	Moment of inertia about O
${}^B\boldsymbol{R}_A$	Orthonormal rotation matrix transforming from A coordinates to B coordinates
${}^A\boldsymbol{p}_B$	Location of the origin of B coordinates relative to the origin of A coordinates, expressed in A coordinates

verify this, we can recover them as follows. Expressing (3.35) at the body's center of mass, and using (3.16), (3.14), and (3.22), we have

$$\begin{aligned}\begin{pmatrix}\underline{\boldsymbol{n}}_C\\ \underline{\boldsymbol{f}}\end{pmatrix} &= \begin{pmatrix}\bar{\boldsymbol{I}}^{\mathrm{cm}} & \boldsymbol{0}\\ \boldsymbol{0} & m\boldsymbol{1}\end{pmatrix}\begin{pmatrix}\dot{\underline{\boldsymbol{\omega}}}\\ \dot{\underline{\boldsymbol{v}}}_C\end{pmatrix} - \begin{pmatrix}S(\underline{\boldsymbol{\omega}})^{\mathrm{T}} & S(\underline{\boldsymbol{v}}_C)^{\mathrm{T}}\\ \boldsymbol{0} & S(\underline{\boldsymbol{\omega}})^{\mathrm{T}}\end{pmatrix}\begin{pmatrix}\bar{\boldsymbol{I}}^{\mathrm{cm}}\underline{\boldsymbol{\omega}}\\ m\underline{\boldsymbol{v}}_C\end{pmatrix}\\ &= \begin{pmatrix}\bar{\boldsymbol{I}}^{\mathrm{cm}} & \boldsymbol{0}\\ \boldsymbol{0} & m\boldsymbol{1}\end{pmatrix}\begin{pmatrix}\dot{\underline{\boldsymbol{\omega}}}\\ \ddot{\underline{\boldsymbol{c}}} - \underline{\boldsymbol{\omega}}\times\underline{\boldsymbol{v}}_C\end{pmatrix} + \begin{pmatrix}\underline{\boldsymbol{\omega}}\times\bar{\boldsymbol{I}}^{\mathrm{cm}}\underline{\boldsymbol{\omega}}\\ m\underline{\boldsymbol{\omega}}\times\underline{\boldsymbol{v}}_C\end{pmatrix}\\ &= \begin{pmatrix}\bar{\boldsymbol{I}}^{\mathrm{cm}}\dot{\underline{\boldsymbol{\omega}}} + \underline{\boldsymbol{\omega}}\times\bar{\boldsymbol{I}}^{\mathrm{cm}}\underline{\boldsymbol{\omega}}\\ m\ddot{\underline{\boldsymbol{c}}}\end{pmatrix}.\end{aligned} \tag{3.36}$$

Table 3.1 (continued)

Equations
$\mathbf{v} = \begin{pmatrix}\boldsymbol{\omega}\\ \boldsymbol{v}_O\end{pmatrix} \quad \mathbf{a} = \begin{pmatrix}\dot{\boldsymbol{\omega}}\\ \dot{\boldsymbol{v}}_O\end{pmatrix} = \begin{pmatrix}\dot{\boldsymbol{\omega}}\\ \ddot{\boldsymbol{r}} - \boldsymbol{\omega}\times\dot{\boldsymbol{r}}\end{pmatrix}$
$\mathbf{f} = \begin{pmatrix}\boldsymbol{n}_O\\ \boldsymbol{f}\end{pmatrix} \quad \mathbf{a}' = \begin{pmatrix}\dot{\boldsymbol{\omega}}\\ \dot{\boldsymbol{v}}'_O\end{pmatrix} = \begin{pmatrix}\dot{\boldsymbol{\omega}}\\ \ddot{\boldsymbol{r}}\end{pmatrix} = \mathbf{a} + \begin{pmatrix}\boldsymbol{0}\\ \boldsymbol{\omega}\times\boldsymbol{v}_O\end{pmatrix}$
$\boldsymbol{I} = \begin{pmatrix}\bar{\boldsymbol{I}} & S(\boldsymbol{h})\\ S(\boldsymbol{h})^{\mathrm{T}} & m\boldsymbol{1}\end{pmatrix} = \begin{pmatrix}\bar{\boldsymbol{I}}^{\mathrm{cm}} + mS(\boldsymbol{c})S(\boldsymbol{c})^{\mathrm{T}} & mS(\boldsymbol{c})\\ mS(\boldsymbol{c})^{\mathrm{T}} & m\boldsymbol{1}\end{pmatrix}$
${}^B\boldsymbol{X}_A = \begin{pmatrix}{}^B\boldsymbol{R}_A & \boldsymbol{0}\\ {}^B\boldsymbol{R}_A S({}^A\boldsymbol{p}_B)^{\mathrm{T}} & {}^B\boldsymbol{R}_A\end{pmatrix} = \begin{pmatrix}{}^B\boldsymbol{R}_A & \boldsymbol{0}\\ S({}^B\boldsymbol{p}_A)\,{}^B\boldsymbol{R}_A & {}^B\boldsymbol{R}_A\end{pmatrix}$
$\mathbf{v}\cdot\mathbf{f} = \mathbf{f}\cdot\mathbf{v} = \mathbf{v}^{\mathrm{T}}\mathbf{f} = \boldsymbol{\omega}\cdot\boldsymbol{n}_O + \boldsymbol{v}_O\cdot\boldsymbol{f}$
$\mathbf{v}\times\mathbf{m} = \begin{pmatrix}\boldsymbol{\omega}\times\boldsymbol{m}\\ \boldsymbol{v}_O\times\boldsymbol{m} + \boldsymbol{\omega}\times\boldsymbol{m}_O\end{pmatrix} = \begin{pmatrix}S(\boldsymbol{\omega}) & \boldsymbol{0}\\ S(\boldsymbol{v}_O) & S(\boldsymbol{\omega})\end{pmatrix}\begin{pmatrix}\boldsymbol{m}\\ \boldsymbol{m}_O\end{pmatrix}$
$\mathbf{v}\times\mathbf{f} = \begin{pmatrix}\boldsymbol{\omega}\times\boldsymbol{n}_O + \boldsymbol{v}_O\times\boldsymbol{f}\\ \boldsymbol{\omega}\times\boldsymbol{f}\end{pmatrix} = \begin{pmatrix}S(\boldsymbol{\omega}) & S(\boldsymbol{v}_O)\\ \boldsymbol{0} & S(\boldsymbol{\omega})\end{pmatrix}\begin{pmatrix}\boldsymbol{n}_O\\ \boldsymbol{f}\end{pmatrix}$

Compact computer representations			
Mathematical object	**Size**	**Computer representation**	**Size**
$\begin{pmatrix}\boldsymbol{\omega}\\ \boldsymbol{v}_O\end{pmatrix}$	6×1	$(\boldsymbol{\omega}\,;\,\boldsymbol{v}_O)$	3+3
$\begin{pmatrix}\boldsymbol{n}_O\\ \boldsymbol{f}\end{pmatrix}$	6×1	$(\boldsymbol{n}_O\,;\,\boldsymbol{f})$	3+3
$\begin{pmatrix}\bar{\boldsymbol{I}} & S(\boldsymbol{h})\\ S(\boldsymbol{h})^{\mathrm{T}} & m\boldsymbol{1}\end{pmatrix}$	6×6	$(m\,;\,\boldsymbol{h}\,;\,\bar{\boldsymbol{I}})$	1+3+9
$\begin{pmatrix}\boldsymbol{R} & \boldsymbol{0}\\ \boldsymbol{R}S(\boldsymbol{p})^{\mathrm{T}} & \boldsymbol{R}\end{pmatrix}$	6×6	$(\boldsymbol{R}\,;\,\boldsymbol{p})$	9+3

Efficient spatial arithmetic formulae	
Expression	**Computed value**
$\boldsymbol{X}\,\mathbf{v}$	$(\boldsymbol{R}\,\boldsymbol{\omega}\,;\,\boldsymbol{R}(\boldsymbol{v}_O - \boldsymbol{p}\times\boldsymbol{\omega}))$
$\boldsymbol{X}^F\,\mathbf{f}$	$(\boldsymbol{R}(\boldsymbol{n}_O - \boldsymbol{p}\times\boldsymbol{f})\,;\,\boldsymbol{R}\boldsymbol{f})$
$\boldsymbol{X}^{-1}$	$(\boldsymbol{R}^{\mathrm{T}}\,;\,-\boldsymbol{R}\boldsymbol{p})$
$\boldsymbol{X}^{-1}\,\mathbf{v}$	$(\boldsymbol{R}^{\mathrm{T}}\boldsymbol{\omega}\,;\,\boldsymbol{R}^{\mathrm{T}}\boldsymbol{v}_O + \boldsymbol{p}\times\boldsymbol{R}^{\mathrm{T}}\boldsymbol{\omega})$
$(\boldsymbol{X}^F)^{-1}\,\mathbf{f}$	$(\boldsymbol{R}^{\mathrm{T}}\boldsymbol{n}_O + \boldsymbol{p}\times\boldsymbol{R}^{\mathrm{T}}\boldsymbol{f}\,;\,\boldsymbol{R}^{\mathrm{T}}\boldsymbol{f})$
$\boldsymbol{X}_1\boldsymbol{X}_2$	$(\boldsymbol{R}_1\boldsymbol{R}_2\,;\,\boldsymbol{p}_2 + \boldsymbol{R}_2^{\mathrm{T}}\boldsymbol{p}_1)$
$\boldsymbol{I}_1 + \boldsymbol{I}_2$	$(m_1 + m_2\,;\,\boldsymbol{h}_1 + \boldsymbol{h}_2\,;\,\bar{\boldsymbol{I}}_1 + \bar{\boldsymbol{I}}_2)$
$\boldsymbol{I}\,\mathbf{v}$	$(\bar{\boldsymbol{I}}\boldsymbol{\omega} + \boldsymbol{h}\times\boldsymbol{v}_O\,;\,m\,\boldsymbol{v}_O - \boldsymbol{h}\times\boldsymbol{\omega})$
$\boldsymbol{X}^{\mathrm{T}}\boldsymbol{I}\boldsymbol{X}$	$(m\,;\,\boldsymbol{R}^{\mathrm{T}}\boldsymbol{h} + m\boldsymbol{p}\,;\,\boldsymbol{R}^{\mathrm{T}}\bar{\boldsymbol{I}}\boldsymbol{R} - S(\boldsymbol{p})S(\boldsymbol{R}^{\mathrm{T}}\boldsymbol{h}) - S(\boldsymbol{R}^{\mathrm{T}}\boldsymbol{h} + m\boldsymbol{p})S(\boldsymbol{p}))$

For meaning of $\boldsymbol{X}^{\mathrm{T}}\boldsymbol{I}\boldsymbol{X}$ see (3.32)

3.2.13 Computer Implementation

The easiest way to implement spatial vector arithmetic on a computer is to start with an existing matrix arithmetic tool, like MATLAB and write (or download from the Web) routines to do the following:

1. Calculate $\boldsymbol{S}(\mathbf{m})$ from $\mathbf{m}$ according to (3.14).
2. Compose $\boldsymbol{X}$ from $\boldsymbol{R}$ and $\boldsymbol{p}$ according to (3.9).
3. Compose $\boldsymbol{I}$ from m, $\boldsymbol{c}$, and $\bar{\boldsymbol{I}}^{\mathrm{cm}}$ according to (3.28).

All other spatial arithmetic operations can be performed using standard matrix arithmetic routines. However, some additional routines could usefully be added to this list, such as:

- Routines to calculate $\boldsymbol{R}$ from various other representations of rotation.
- Routines to convert between spatial and 4×4 matrix quantities.

This is the recommended approach whenever human productivity is more important than computational efficiency. A software package along these lines can be found at [3.16].

If greater efficiency is required, then a more elaborate spatial arithmetic library must be used, in which

1. A dedicated data structure is defined for each kind of spatial quantity, and
2. A suite of calculation routines are provided, each implementing a spatial arithmetic operation by means of an efficient formula.

Some examples of suitable data structures and efficient formulae are shown in Table 3.1. Observe that the suggested data structures for rigid-body inertias and Plücker transforms contain only a third as many numbers as the 6×6 matrices they represent. The efficient arithmetic formulae listed in this table offer cost savings ranging from a factor of 1.5 to a factor of 6 relative to the use of general 6×6 and 6×1 matrix arithmetic. Even more efficient formulae can be found in [3.17].

3.2.14 Summary

Spatial vectors are 6-D vectors that combine the linear and angular aspects of rigid-body motion, resulting in a compact notation that is very suitable for describing dynamics algorithms. To avoid a few name clashes with 3-D vectors, we have used bold upright letters to denote spatial vectors, while tensors are still denoted by italics. In the sections that follow, upright letters will be used to denote both spatial vectors and vectors that are concatenations of other vectors, like $\dot{\mathbf{q}}$.

Table 3.1 presents a summary of the spatial quantities and operators introduced in this section, together with the formulae that define them in terms of 3-D quantities and operators. It also presents data structures and formulae for efficient computer implementation of spatial arithmetic. This table should be read in conjunction with Tables 2.5 and 2.6, which show how to compute the orientation, position, and spatial velocities for a variety of joint types. Note that ${}^{j}\boldsymbol{R}_{i}$ and ${}^{j}\boldsymbol{p}_{i}$ in these tables correspond to ${}^{B}\boldsymbol{R}_{A}^{T}$ and ${}^{A}\boldsymbol{p}_{B}$, respectively, when read in conjunction with Table 3.1.

3.3 Canonical Equations

The equations of motion of a robot mechanism are usually presented in one of two canonical forms: the joint-space formulation,

$$\boldsymbol{H}(\mathbf{q})\ddot{\mathbf{q}}+\boldsymbol{C}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}}+\boldsymbol{\tau}_{\mathrm{g}}(\mathbf{q})=\boldsymbol{\tau}\ , \tag{3.37}$$

or the operational-space formulation,

$$\boldsymbol{\Lambda}(\boldsymbol{x})\dot{\mathbf{v}}+\boldsymbol{\mu}(\boldsymbol{x},\mathbf{v})+\boldsymbol{\rho}(\boldsymbol{x})=\mathbf{f}\,. \tag{3.38}$$

These equations show the functional dependencies explicitly: $\boldsymbol{H}$ is a function of $\mathbf{q}$, $\boldsymbol{\Lambda}$ is a function of $\boldsymbol{x}$, and so on. Once these dependencies are understood, they are usually omitted. In (3.38), $\boldsymbol{x}$ is a vector of operational-space coordinates, while $\mathbf{v}$ and $\mathbf{f}$ are spatial vectors denoting the velocity of the end-effector and the external force acting on it. If the robot is redundant, then the coefficients of this equation must be defined as functions of $\mathbf{q}$ and $\dot{\mathbf{q}}$ rather than $\boldsymbol{x}$ and $\mathbf{v}$.

These two equations are further explained below, along with a description of the Lagrange formulation of (3.37), and the impulsive equations of motion for impact.

3.3.1 Joint-Space Formulation

The symbols $\mathbf{q}$, $\dot{\mathbf{q}}$, $\ddot{\mathbf{q}}$, and $\boldsymbol{\tau}$ denote n-dimensional vectors of joint position, velocity, acceleration and force variables, respectively, where n is the number of degrees of motion freedom of the robot mechanism. $\boldsymbol{H}$ is an $n\times n$ symmetric, positive-definite matrix, and is called the generalized, or joint-space, inertia matrix (JSIM). $\boldsymbol{C}$ is an $n\times n$ matrix such that $\boldsymbol{C}\,\dot{\mathbf{q}}$ is the vector of Coriolis and centrifugal terms (collectively known as *velocity product* terms); and $\boldsymbol{\tau}_{\mathrm{g}}$ is the vector of gravity terms. More terms can be added to this equation, as required, to account for other dynamical effects (e.g., viscous friction). The effects of a force $\mathbf{f}$ exerted on the mechanism at the end-effector can be accounted for by adding the term $\boldsymbol{J}^{\mathrm{T}}\mathbf{f}$ to the right side of (3.37), where $\boldsymbol{J}$ is the Jacobian of the end-effector (Sect. 2.8.1).

$\mathbf{q}$ specifies the coordinates of a point in the mechanism's configuration space. If the mechanism is a kinematic tree (Sect. 3.4), then $\mathbf{q}$ contains every joint variable in the mechanism, otherwise it contains only an independent subset. The elements of $\mathbf{q}$ are generalized coordinates. Likewise, the elements of $\dot{\mathbf{q}}$, $\ddot{\mathbf{q}}$, and $\boldsymbol{\tau}$ are generalized velocities, accelerations, and forces.

3.3.2 Lagrange Formulation

Various methods exist for deriving the terms in (3.37). The two that are most commonly used in robotics are the Newton–Euler formulation and the Lagrange formulation. The former works directly with Newton's and Euler's equations for a rigid body, which are contained within the spatial equation of motion, (3.35). This formulation is especially amenable to the development of efficient recursive algorithms for dynamics computations, such as those described in Sects. 3.5 and 3.6.

The Lagrange formulation proceeds via the Lagrangian of the robot mechanism,

$$L = T - U \,, \tag{3.39}$$

where T and U are the total kinetic and potential energy, respectively, of the mechanism. The kinetic energy is given by

$$T = \frac{1}{2}\dot{\mathbf{q}}^{\mathrm{T}}\boldsymbol{H}\dot{\mathbf{q}} \,. \tag{3.40}$$

The dynamic equations of motion can then be developed using Lagrange's equation for each generalized coordinate

$$\frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial L}{\partial \dot{q}_i} - \frac{\partial L}{\partial q_i} = \tau_i \,. \tag{3.41}$$

The resulting equation can be written in scalar form

$$\sum_{j=1}^{n} H_{ij}\ddot{q}_j + \sum_{j=1}^{n}\sum_{k=1}^{n} C_{ijk}\dot{q}_j\dot{q}_k + \tau_{gi} = \tau_i \,, \tag{3.42}$$

which shows the structure of the velocity-product terms. C_{ijk} are known as Christoffel symbols of the first type, and are given by

$$C_{ijk} = \frac{1}{2}\left(\frac{\partial H_{ij}}{\partial q_k} + \frac{\partial H_{ik}}{\partial q_j} - \frac{\partial H_{jk}}{\partial q_i}\right) \,. \tag{3.43}$$

They are functions of only the position variables, q_i. The elements of $\boldsymbol{C}$ in (3.37) can be defined as

$$C_{ij} = \sum_{k=1}^{n} C_{ijk}\dot{q}_k \,. \tag{3.44}$$

However, $\boldsymbol{C}$ is not unique, and other definitions are possible.

With the choice of $\boldsymbol{C}$ given in (3.44), it is possible to show that the matrix $\boldsymbol{N}$, given by

$$\boldsymbol{N}(\mathbf{q},\dot{\mathbf{q}}) = \dot{\boldsymbol{H}}(\mathbf{q}) - 2\boldsymbol{C}(\mathbf{q},\dot{\mathbf{q}}) \,, \tag{3.45}$$

is skew-symmetric [3.18]. Thus, for any $n \times 1$ vector $\boldsymbol{\alpha}$,

$$\boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{N}(\mathbf{q},\dot{\mathbf{q}})\,\boldsymbol{\alpha} = 0 \,. \tag{3.46}$$

This property is quite useful in control, especially when considering $\boldsymbol{\alpha} = \dot{\mathbf{q}}$, which gives

$$\dot{\mathbf{q}}^{\mathrm{T}}\boldsymbol{N}(\mathbf{q},\dot{\mathbf{q}})\,\dot{\mathbf{q}} = 0 \,. \tag{3.47}$$

By applying the principle of conservation of energy, it can be shown that (3.47) holds for any choice of the matrix $\boldsymbol{C}$ [3.18, 19].

3.3.3 Operational-Space Formulation

In (3.38), $\boldsymbol{x}$ is a 6-D vector of operational-space coordinates giving the position and orientation of the robot's end-effector; $\mathbf{v}$ is the velocity of the end-effector; and $\mathbf{f}$ is the force exerted on the end-effector. $\boldsymbol{x}$ is typically a list of Cartesian coordinates, and Euler angles or quaternion components, and is related to $\mathbf{v}$ via a differential equation of the form

$$\dot{\boldsymbol{x}} = \boldsymbol{E}(\boldsymbol{x})\mathbf{v} \,. \tag{3.48}$$

$\boldsymbol{\Lambda}$ is the operational-space inertia matrix, which is the apparent inertia of the end-effector taking into account the effect of the rest of the robot's mechanism (i. e., it is an articulated-body inertia). $\boldsymbol{\mu}$ and $\boldsymbol{\rho}$ are vectors of velocity-product and gravity terms, respectively.

Operational space (also known as task space) is the space in which high-level motion and force commands are issued and executed. The operational-space formulation is therefore particularly useful in the context of motion and force control systems (Sects. 8.2 and 9.2). Equation (3.38) can be generalized to operational spaces with dimensions other than six, and to operational spaces that incorporate the motions of more than one end-effector [3.20].

The terms in (3.37) and (3.38) are related by the following formulae

$$\mathbf{v} = \boldsymbol{J}\dot{\mathbf{q}} \,, \tag{3.49}$$

$$\dot{\mathbf{v}} = \boldsymbol{J}\ddot{\mathbf{q}} + \dot{\boldsymbol{J}}\dot{\mathbf{q}} \,, \tag{3.50}$$

$$\boldsymbol{\tau} = \boldsymbol{J}^{\mathrm{T}}\mathbf{f} \,, \tag{3.51}$$

$$\boldsymbol{\Lambda} = (\boldsymbol{J}\boldsymbol{H}^{-1}\boldsymbol{J}^{\mathrm{T}})^{-1} \,, \tag{3.52}$$

$$\boldsymbol{\mu} = \boldsymbol{\Lambda}(\boldsymbol{J}\boldsymbol{H}^{-1}\boldsymbol{C}\dot{\mathbf{q}} - \dot{\boldsymbol{J}}\dot{\mathbf{q}}) \,, \tag{3.53}$$

and

$$\boldsymbol{\rho} = \boldsymbol{\Lambda}\boldsymbol{J}\boldsymbol{H}^{-1}\boldsymbol{\tau}_{\mathrm{g}} \,. \tag{3.54}$$

These equations assume that $m \leq n$ (m is the dimension of operational-space coordinates), and that the Jacobian $\boldsymbol{J}$ has full rank. More details can be found in [3.21].

3.3.4 Impact Model

If a robot strikes a rigid body in its environment, then an impulsive force arises at the moment of impact and causes a step change in the robot's velocity. Let us assume that the impact occurs between the end effector and a rigid body in the environment, and that a spatial impulse of $\mathbf{f}'$ is exerted on the end-effector. This impulse causes a step change of $\Delta\mathbf{v}$ in the end-effector's velocity; and the two are related by the operational-space equation of impulsive motion [3.22],

$$\boldsymbol{\Lambda}\Delta\mathbf{v} = \mathbf{f}' \,. \tag{3.55}$$

In joint space, the equation of impulsive motion for a robot mechanism is

$$\boldsymbol{H}\Delta\dot{\mathbf{q}} = \boldsymbol{\tau}' \,, \tag{3.56}$$

where $\boldsymbol{\tau}'$ and $\Delta\dot{\mathbf{q}}$ denote the joint-space impulse and velocity change, respectively. In the case of a collision involving the robot's end-effector, we have

$$\boldsymbol{\tau}' = \boldsymbol{J}^{\mathrm{T}}\mathbf{f}' \tag{3.57}$$

and

$$\Delta\mathbf{v} = \boldsymbol{J}\Delta\dot{\mathbf{q}} \,, \tag{3.58}$$

which follow from (3.51) and (3.49). Equations (3.55)–(3.57) imply that

$$\Delta\dot{\mathbf{q}} = \bar{\boldsymbol{J}}\Delta\mathbf{v} \,, \tag{3.59}$$

where $\bar{\boldsymbol{J}}$ is the inertia-weighted pseudoinverse of $\boldsymbol{J}$ and is given by

$$\bar{\boldsymbol{J}} = \boldsymbol{H}^{-1}\boldsymbol{J}^{\mathrm{T}}\boldsymbol{\Lambda} \,. \tag{3.60}$$

$\bar{\boldsymbol{J}}$ is also known as the dynamically consistent generalized inverse of the Jacobian matrix [3.21]. Note that the expression $\boldsymbol{\Lambda}\boldsymbol{J}\boldsymbol{H}^{-1}$, which appears in (3.53) and (3.54), is equal to $\bar{\boldsymbol{J}}^{\mathrm{T}}$ since $\boldsymbol{H}$ and $\boldsymbol{\Lambda}$ are both symmetric. Although we have introduced $\bar{\boldsymbol{J}}$ in the context of impulsive dynamics, it is more typically used in normal (i. e., non-impulsive) dynamics equations.

3.4 Dynamic Models of Rigid-Body Systems

A basic rigid-body model of a robot mechanism has four components: a connectivity graph, link and joint geometry parameters, link inertia parameters, and a set of joint models. To this model, one can add various force-producing elements, such as springs, dampers, joint friction, actuators, and drives. The actuators and drives, in particular, may have quite elaborate dynamic models of their own. It is also possible to add extra motion freedoms to model elasticity in the joint bearings or links (Chap. 11). This section describes a basic model. More on this topic can be found in books such as [3.3, 8, 23].

3.4.1 Connectivity

A connectivity graph is an undirected graph in which each node represents a rigid body and each arc represents a joint. The graph must be connected; and exactly one node represents a fixed base or reference frame. If the graph represents a mobile robot (i. e., a robot that is not connected to a fixed base), then it is necessary to introduce a fictitious 6-DOF joint between the fixed base and any one body in the mobile robot. The chosen body is then known as a *floating base*. If a single graph is to represent a collection of mobile robots, then each robot has its own floating base, and each floating base has its own 6-DOF joint. Note that a 6-DOF joint imposes no constraints on the two bodies it connects, so the introduction of a 6-DOF joint alters the connectivity of the graph without altering the physical properties of the system it represents.

In graph-theory terminology, a loop is an arc that connects a node to itself, and a cycle is a closed path that does not traverse any arc more than once. In the connectivity graph of a robot mechanism, loops are not allowed, and cycles are called kinematic loops. A mechanism that contains kinematic loops is called a closed-loop mechanism; and a mechanism that does not is called an open-loop mechanism or a kinematic tree. Every closed-loop mechanism has a spanning tree, which defines an open-loop mechanism, and every joint that is not in the spanning tree is called a loop-closing joint. The joints in the tree are called tree joints.

The fixed base serves as the root node of a kinematic tree, and the root node of any spanning tree on a closed-loop mechanism. A kinematic tree is said to be branched if at least one node has at least two children, and unbranched otherwise. An unbranched kinematic

tree is also called a kinematic chain, and a branched tree can be called a branched kinematic chain. A typical industrial robot arm, without a gripper, is a kinematic chain, while a typical humanoid robot is a kinematic tree with a floating base.

In a system containing N_B moving bodies and N_J joints, where N_J includes the 6-DOF joints mentioned above, the bodies and joints are numbered as follows. First, the fixed base is numbered body 0. The other bodies are then numbered from 1 to N_B in any order such that each body has a higher number than its parent. If the system contains kinematic loops then one must first choose a spanning tree, and commit to that choice, since the identity of a body's parent is determined by the spanning tree. This style of numbering is called a *regular numbering scheme*.

Having numbered the bodies, we number the tree joints from 1 to N_B such that joint i connects body i to its parent. The loop-closing joints, if any, are then numbered from N_B+1 to N_J in any order. Each loop-closing joint k closes one independent kinematic loop, and we number the loops from 1 to N_L (where $N_L = N_J - N_B$ is the number of independent loops) such that loop l is the one closed by joint $k = N_B + l$. Kinematic loop l is the unique cycle in the graph that traverses joint k, but does not traverse any other loop-closing joint.

For an unbranched kinematic tree, these rules produce a unique numbering in which the bodies are numbered consecutively from base to tip, and the joints are numbered such that joint i connects bodies i and $i-1$. In all other cases, regular numberings are not unique.

Although the connectivity graph is undirected, it is necessary to assign a direction to each joint for the purpose of defining joint velocity and force. This is necessary for both tree joints and loop-closing joints. Specifically, a joint is said to connect from one body to another. We may call them the predecessor $p(i)$ and successor $s(i)$ for joint i, respectively. Joint velocity is then defined as the velocity of the successor relative to the predecessor; and joint force is defined as a force acting on the successor. It is standard practice (but not a necessity) for all tree joints to connect from the parent to the child.

The connectivity of a kinematic tree, or the spanning tree on a closed-loop mechanism, is described by an N_B-element array of parent body numbers, where the i-th element $p(i)$ is the parent of body i. Note that the parent $p(i)$ for body i is also the predecessor $p(i)$ for joint i, and thus the common notation. Many algorithms rely on the property $p(i) < i$ to perform their calculations in the correct order. The set of all body numbers for the children of body i, $c(i)$, is also useful in many recursive algorithms.

The connectivity data for kinematic loops may be described in a variety of ways. A representation that facilitates use in recursive algorithms includes the following conventions. Loop-closing joint k joins bodies $p(k)$ (the predecessor) and $s(k)$ (the successor). The set $LR(i)$ for body i gives the numbers of the loops for which body i is the root. Using the property $p(i) < i$ for bodies in the spanning tree, the root of a loop is chosen as the body with the lowest number. In addition, the set $LB(i)$ for body i gives the numbers of the loops to which body i belongs but is not the root.

An example of a closed-loop system is given in Fig. 3.3. The system consists of a humanoid mechanism with topologically varying contacts, with the environment and within the mechanism, which form closed loops. The system has $N_B = 16$ moving bodies and $N_J = 19$ joints with $N_L = N_J - N_B = 3$ loops. The main body (1) is considered to be a *floating base* for this mobile robot system. It is connected to the fixed base (0) through a fictitious 6-DOF joint (1). To complete the example, the loop-closing joint and the body numbers $p(k)$ and $s(k)$ as well as the root body for each loop are given in Table 3.2. The body-based sets $c(i)$ and

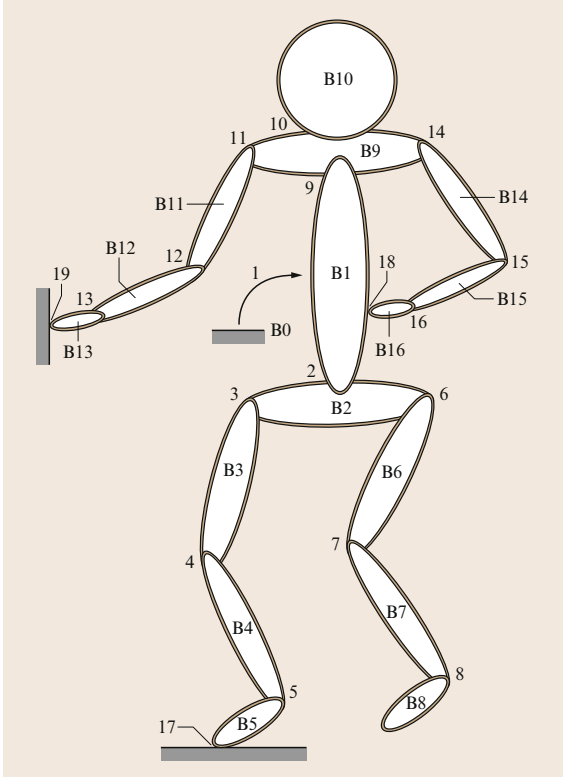


Fig. 3.3 Humanoid robot example. Note: to distinguish between body numbers and joint numbers in this figure, body numbers are preceded by a B for clarity

$LB(i)$ are given in Table 3.3. Note that $LR(0) = \{1, 3\}$ and $LR(1) = \{2\}$ and all other LR sets are null for this example.

3.4.2 Link Geometry

When two bodies are connected by a joint, a complete description of the connection consists of a description of the joint itself, and the locations of two coordinate frames, one in each body, which specify where in each body the joint is located. If there are N_J joints in the system, then there are a total of $2N_J$ joint-attachment frames. One half of these frames are identified with the numbers 1 to N_J, and the remainder with the labels $J1$ to JN_J. Each joint i connects from frame Ji to frame i.

For joints 1 to N_B (i. e., the tree joints), frame i is rigidly attached to body i. For joints $N_B + 1$ to N_J, frame k for loop-closing joint k will be rigidly attached to body $s(k)$. The second coordinate frame Ji is attached to the predecessor $p(i)$ for each joint i, whether it is a tree joint or a loop-closing joint. Coordinate frame Ji provides a base frame for joint i in that the joint rotation and/or translation is defined relative to this frame.

Figure 3.4 shows the coordinate frames and transforms associated with each joint in the system. The overall transform from frame $p(i)$ coordinates to frame i coordinates for a tree joint is given by

$$ {}^{i}\boldsymbol{X}_{p(i)} = {}^{i}\boldsymbol{X}_{Ji}\,{}^{Ji}\boldsymbol{X}_{p(i)} = \boldsymbol{X}_J(i)\boldsymbol{X}_L(i) \,. \tag{3.61} $$

The transform $\boldsymbol{X}_L(i)$ is a fixed link transform which sets the base frame Ji of joint i relative to $p(i)$. It may be used to transform spatial motion vectors from $p(i)$ to Ji coordinates. The transform $\boldsymbol{X}_J(i)$ is a variable joint transform which completes the transformation across joint i from Ji to i coordinates.

Similarly, the overall transform from frame $p(k)$ coordinates to frame k coordinates for a loop-closing joint is given by

$$ {}^{k}\boldsymbol{X}_{p(k)} = {}^{k}\boldsymbol{X}_{Jk}\,{}^{Jk}\boldsymbol{X}_{p(k)} = \boldsymbol{X}_J(k)\,\boldsymbol{X}_{L1}(k) \,. \tag{3.62} $$

An additional transform $\boldsymbol{X}_{L2}(k)$ is defined from frame $s(k)$ coordinates to frame k coordinates and is given by

$$ \boldsymbol{X}_{L2}(k) = {}^{k}\boldsymbol{X}_{s(k)} \,. \tag{3.63} $$

Link and joint geometry data can be specified in a variety of different ways. The most common method is to use *Denavit–Hartenberg* parameters [3.24]. However, standard Denavit–Hartenberg parameters are not completely general, and are insufficient for describing the geometry for a branched kinematic tree, or for a mechanism containing certain kinds of multi-DOF joints. A modified form of Denavit–Hartenberg parameters [3.2] is used for single-DOF joints in this Handbook (Sect. 2.4). The parameters have been extended for branched kinematic trees [3.23] and closed-loop mechanisms.

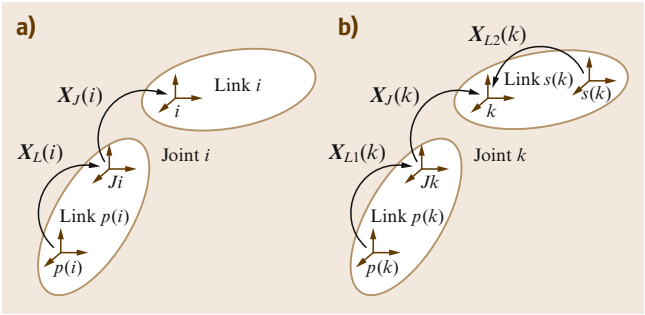


Fig.3.4a,b Coordinate frames and transforms associated with **(a)** a tree joint and **(b)** a loop-closing joint

Table 3.2 Loop-closing joints and roots of loops for the humanoid example

Loop l	Loop-closing joint k	$p(k)$	$s(k)$	Root
1	17	0	5	0
2	18	16	1	1
3	19	0	13	0

Table 3.3 Body-based sets for the humanoid example

Body i	$c(i)$	$LB(i)$	Body i	$c(i)$	$LB(i)$
0	1		9	10, 11, 14	2, 3
1	2, 9	1, 3	10		
2	3, 6	1	11	12	3
3	4	1	12	13	3
4	5	1	13		3
5		1	14	15	2
6	7		15	16	2
7	8		16		2
8					

3.4.3 Link Inertias

The link inertia data consists of the masses, positions of centers of mass, and rotational inertias of each link in the mechanism. The inertia parameters for link i are expressed in coordinate frame i, and are therefore constants.

3.4.4 Joint Models

The relationship between connected links is described using the general joint model of *Roberson* and *Schwertassek* [3.3]. For a kinematic tree or spanning tree on

a closed-loop mechanism, an $n_i \times 1$ vector, $\dot{\boldsymbol{q}}_i$, relates the velocity of link i to the velocity of its parent, link $p(i)$, where n_i is the number of degrees of freedom at the joint connecting the two links. For a loop-closing joint in a closed-loop mechanism, the relationship is between the velocity of link $s(i)$ (the successor) and the velocity of link $p(i)$ (the predecessor). In either case, the relationship is between the velocity of coordinate frames i and Ji.

Let $\mathbf{v}_{\text{rel}}$ and $\mathbf{a}_{\text{rel}}$ denote the velocity and acceleration across joint i, that is, the velocity and acceleration of link $s(i)$ relative to $p(i)$. The free modes of the joint are represented by the $6 \times n_i$ matrix $\boldsymbol{\Phi}_i$, such that $\mathbf{v}_{\text{rel}}$ and $\mathbf{a}_{\text{rel}}$ are given as follows

$$\mathbf{v}_{\text{rel}} = \boldsymbol{\Phi}_i \dot{\boldsymbol{q}}_i \tag{3.64}$$

and

$$\mathbf{a}_{\text{rel}} = \boldsymbol{\Phi}_i \ddot{\boldsymbol{q}}_i + \dot{\boldsymbol{\Phi}}_i \dot{\boldsymbol{q}}_i \,, \tag{3.65}$$

where $\boldsymbol{\Phi}_i$ and $\dot{\boldsymbol{\Phi}}_i$ depend on the type of joint [3.3]. The matrix $\boldsymbol{\Phi}_i$ has full column rank, so we can define a complementary matrix, $\boldsymbol{\Phi}_i^{\text{c}}$, such that the 6×6 matrix $\left(\boldsymbol{\Phi}_i \boldsymbol{\Phi}_i^{\text{c}}\right)$ is invertible. We can regard the columns of this matrix as forming a basis on M^6 such that the first n_i basis vectors define the directions in which motion is allowed, and the remaining $6 - n_i = n_i^{\text{c}}$ vectors define directions in which motion is not allowed. Thus, $\boldsymbol{\Phi}_i^{\text{c}}$ represents the constrained modes of joint i.

The force transmitted across joint i from its predecessor to its successor, $\mathbf{f}_i$, is given as follows

$$\mathbf{f}_i = \left(\boldsymbol{\Psi}_i \boldsymbol{\Psi}_i^{\text{c}}\right) \begin{pmatrix} \boldsymbol{\tau}_i \\ \boldsymbol{\lambda}_i \end{pmatrix} , \tag{3.66}$$

where $\boldsymbol{\tau}_i$ is the $n_i \times 1$ vector of applied forces along the free modes, $\boldsymbol{\lambda}_i$ is the $(6 - n_i) \times 1$ vector of constraint forces, and $\boldsymbol{\Psi}_i$ and $\boldsymbol{\Psi}_i^{\text{c}}$ are computed as follows

$$\left(\boldsymbol{\Psi}_i \boldsymbol{\Psi}_i^{\text{c}}\right) = \left(\boldsymbol{\Phi}_i \boldsymbol{\Phi}_i^{\text{c}}\right)^{-T} . \tag{3.67}$$

For most common joint types, it is possible to choose $\boldsymbol{\Phi}_i$ and $\boldsymbol{\Phi}_i^{\text{c}}$ such that the matrix $(\boldsymbol{\Phi}_i \boldsymbol{\Phi}_i^{\text{c}})$ is numerically orthonormal, so that $(\boldsymbol{\Psi}_i \boldsymbol{\Psi}_i^{\text{c}})$ is numerically equal to $(\boldsymbol{\Phi}_i \boldsymbol{\Phi}_i^{\text{c}})$. Note that (3.67) implies the following relationships: $(\boldsymbol{\Psi}_i)^{\text{T}} \boldsymbol{\Phi}_i = \mathbf{1}_{n_i \times n_i}$, $(\boldsymbol{\Psi}_i)^{\text{T}} \boldsymbol{\Phi}_i^{\text{c}} = \mathbf{0}_{n_i \times (6-n_i)}$, $(\boldsymbol{\Psi}_i^{\text{c}})^{\text{T}} \boldsymbol{\Phi}_i = \mathbf{0}_{(6-n_i) \times n_i}$, and $(\boldsymbol{\Psi}_i^{\text{c}})^{\text{T}} \boldsymbol{\Phi}_i^{\text{c}} = \mathbf{1}_{(6-n_i) \times (6-n_i)}$. When applied to (3.66), the following useful relationship results

$$\boldsymbol{\tau}_i = \boldsymbol{\Phi}_i^{\text{T}} \mathbf{f}_i \,. \tag{3.68}$$

The value of $\dot{\boldsymbol{\Phi}}_i$ in (3.65) depends on the type of joint. The general formula is

$$\dot{\boldsymbol{\Phi}}_i = \mathring{\boldsymbol{\Phi}}_i + \mathbf{v}_i \times \boldsymbol{\Phi}_i \,, \tag{3.69}$$

where $\mathbf{v}_i$ is the velocity of link i, and $\mathring{\boldsymbol{\Phi}}_i$ is the apparent derivative of $\boldsymbol{\Phi}_i$, as seen by an observer moving with link i, and is given by

$$\mathring{\boldsymbol{\Phi}}_i = \frac{\partial \boldsymbol{\Phi}_i}{\partial \boldsymbol{q}_i} \dot{\boldsymbol{q}}_i \,. \tag{3.70}$$

For most common joint types, $\mathring{\boldsymbol{\Phi}}_i = \mathbf{0}$.

Single-DOF joints ($n_i = 1$) are especially straightforward to work with when using the Denavit–Hartenberg convention. Motion is chosen along (prismatic) or about (revolute) the $\hat{\boldsymbol{z}}_i$ coordinate axis. In this case, $\boldsymbol{\Phi}_i = (0\,0\,0\,0\,0\,1)^{\text{T}}$ for a prismatic joint and $\boldsymbol{\Phi}_i = (0\,0\,1\,0\,0\,0)^{\text{T}}$ for a revolute joint. Also, $\mathring{\boldsymbol{\Phi}}_i = \mathbf{0}$.

The fictitious 6-DOF joint for a floating base for a mobile robot is also handled relatively easily. For this case, $\boldsymbol{\Phi}_i = \mathbf{1}$ (6×6 identity matrix) and $\mathring{\boldsymbol{\Phi}}_i = \mathbf{0}$.

The revolute joint and floating-base joint, as well as the universal joint ($n_i = 2$) and spherical joint ($n_i = 3$) are illustrated in the example in the next section. For additional details on joint kinematics, see Sect. 2.3.

3.4.5 Example System

In order to illustrate the conventions used for the link and joint models, coordinate frames are attached to the first five links (bodies) and fixed base of the humanoid robot as shown in Fig. 3.5. Note that frame Ji is attached to link $p(i) = i - 1$ for each of the five joints. For this example, the origin of frame $J1$ is set coincident with the origin of frame 0, and the origins of frames $J2$, $J3$, $J4$, and $J5$ are coincident with the origins of frames 2, 3, 4, and 5, respectively.

Note that $J1$ could be set at any position/orientation on the fixed base (B0) to permit the most convenient

Table 3.4 Number of degrees of freedom (n_i), fixed rotation $\left({}^{Ji}\boldsymbol{R}_{p(i)}\right)$ and position $\left({}^{p(i)}\boldsymbol{p}_{Ji}\right)$ from frame $p(i)$ to base frame Ji for joint i of the example system. Note that $2l_i$ is the nominal length of link i along its long axis

Joint	n_i	${}^{Ji}\boldsymbol{R}_{p(i)}$	${}^{p(i)}\boldsymbol{p}_{Ji}$
1	6	$\mathbf{1}_{3\times3}$	$\mathbf{0}_{3\times1}$
2	1	$\mathbf{1}_{3\times3}$	$\begin{pmatrix} 0 \\ 0 \\ -l_1 \end{pmatrix}$
3	3	$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{pmatrix}$	$\begin{pmatrix} 0 \\ -l_2 \\ 0 \end{pmatrix}$
4	1	$\mathbf{1}_{3\times3}$	$\begin{pmatrix} 0 \\ 2l_3 \\ 0 \end{pmatrix}$
5	2	$\begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} 0 \\ 2l_4 \\ 0 \end{pmatrix}$

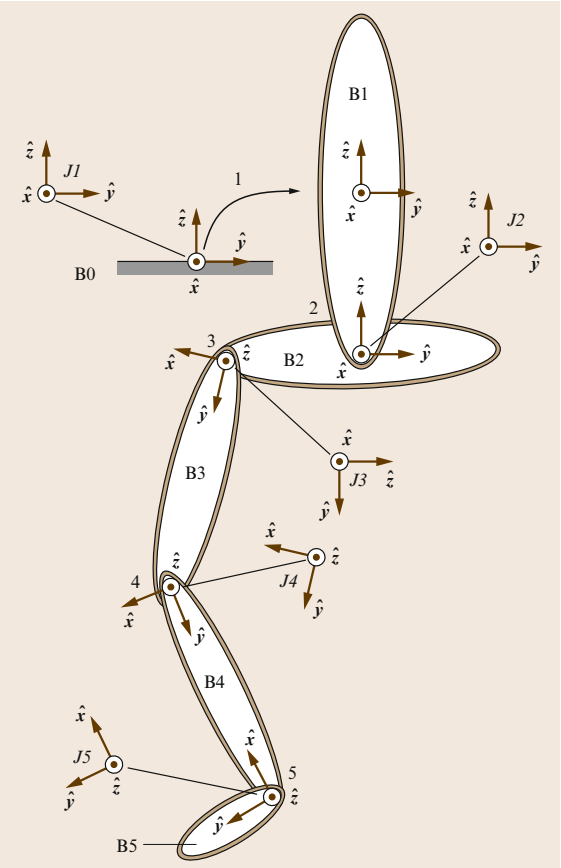


Fig. 3.5 Coordinate frames for the first five links and joints of the humanoid robot example

representation for the motion of the floating base (B1) relative to the fixed base. Also, the origin of $J2$ could be set anywhere along $\hat{\boldsymbol{z}}_2$.

The number of degrees of freedom, and fixed rotation and position of the base frames Ji for each joint of the example system, are given in Table 3.4. The rotation ${}^{Ji}\boldsymbol{R}_{p(i)}$ transforms 3-D vectors in $p(i)$ coordinates to Ji coordinates. The position ${}^{p(i)}\boldsymbol{p}_{Ji}$ is the vector giving the position of the origin O_{Ji} relative to $O_{p(i)}$, expressed in $p(i)$ coordinates. The spatial transform $\boldsymbol{X}_L(i) = {}^{Ji}\boldsymbol{X}_{p(i)}$ may be composed from these 3-D quantities through the equation for ${}^{B}\boldsymbol{X}_A$ in Table 3.1. The humanoid has a floating base, the torso, a revolute joint between the torso and pelvis (about $\hat{\boldsymbol{z}}_2$), a spherical joint at the hip, a revolute joint at the knee, and a universal joint at the ankle. As shown in Fig. 3.5, the leg is slightly bent and the foot is turned out to the side ($\approx 90^\circ$ rotation about $\hat{\boldsymbol{y}}_3$ at the hip).

The free modes, velocity variables, and position variables for all of the joint types in the humanoid are given in Tables 2.5 and 2.6. The expressions for ${}^{j}\boldsymbol{R}_i$ and ${}^{j}\boldsymbol{p}_i$ in these tables give ${}^{i}\boldsymbol{R}_{Ji}^{\mathrm{T}}$ and ${}^{Ji}\boldsymbol{p}_i$, respectively, through which the joint transform $\boldsymbol{X}_J(i) = {}^{i}\boldsymbol{X}_{Ji}$ may be composed. Revolute joints follow the Denavit–Hartenberg convention with rotation about the $\hat{\boldsymbol{z}}_i$ axis. The ankle has a pitch rotation of α_5 about the $\hat{\boldsymbol{z}}_{J5}$ axis followed by a roll rotation of β_5 about the $\hat{\boldsymbol{y}}_5$ axis (see the Z–Y–X Euler angle definitions in Table 2.1). The hip is modeled as a ball-and-socket, spherical joint. To avoid the singularities that are associated with Euler angles, the quaternion $\boldsymbol{\epsilon}_i$ may be used to represent the orientation at the hip. The relationship between the quaternion rate $\dot{\boldsymbol{\epsilon}}_i$ and the relative rotation rate $\boldsymbol{\omega}_{i\,\mathrm{rel}}$ is given in (2.8) of the Handbook.

The floating base uses the position of the torso ${}^{0}\boldsymbol{p}_1$ and quaternion $\boldsymbol{\epsilon}_1$ for its position and orientation state variables, respectively. The position of the torso may be computed by integrating the velocity for the link, as expressed in fixed base coordinates: ${}^{0}\boldsymbol{v}_1 = {}^{0}\boldsymbol{R}_1\boldsymbol{v}_1$, where $\boldsymbol{v}_1$ is the velocity of the torso in moving coordinates.

Note that $\mathring{\boldsymbol{\Phi}}_i = 0$ for all joints except the universal joint. Since the components of $\hat{\boldsymbol{z}}_{J5}$ in link 5 coordinates vary with β_5, $\mathring{\hat{\boldsymbol{z}}}_{J5} \neq 0$. See Sect. 2.3 of the Handbook for further details of the joint kinematics.

3.5 Kinematic Trees

The dynamics of a kinematic tree is simpler, and easier to calculate, than the dynamics of a closed-loop mechanism. Indeed, many algorithms for closed-loop mechanisms work by first calculating the dynamics of a spanning tree, and then subjecting it to the loop-closure constraints.

This section describes the following dynamics algorithms for kinematic trees: the recursive Newton–Euler algorithm (RNEA) for inverse dynamics, the articulated-body algorithm (ABA) for forward dynamics, the composite-rigid-body algorithm (CRBA) for calculating the joint-space inertia matrix (JSIM), and two algorithms to calculate the operational-space inertia matrix (OSIM). Implementations of the first three can be found in [3.16].

3.5.1 The Recursive Newton–Euler Algorithm

This is an $O(n)$ algorithm for calculating the inverse dynamics of a fixed-base kinematic tree, and is based on the very efficient RNEA of *Luh* et al. [3.4]. A floating-

base version can be found in [3.8, 15]. Given the joint position and velocity variables, this algorithm calculates the applied joint torque/force variables required to produce a given set of joint accelerations.

The link velocities and accelerations are first computed through an outward recursion from the fixed base to the leaf links of the tree. The required forces on each link are computed using the Newton–Euler equations (3.35) during this recursion. A second, inward recursion uses the force balance equations at each link to compute the spatial force across each joint and the value of each joint torque/force variable. The key step for computational efficiency is to refer most quantities to local link coordinates. Also, the effects of gravity on each link are efficiently included in the equations by accelerating the base of the mechanism upward.

The calculation proceeds in four steps, as follows, with two steps in each of the two recursions.

Step 1

Calculate the velocity and acceleration of each link in turn, starting with the known velocity and acceleration of the fixed base, and working towards the tips (i. e., the leaf nodes in the connectivity graph).

The velocity of each link in a kinematic tree is given by the recursive formula

$$\mathbf{v}_i = \mathbf{v}_{p(i)} + \boldsymbol{\Phi}_i \dot{\boldsymbol{q}}_i \,, \quad (\mathbf{v}_0 = \mathbf{0}) \,, \tag{3.71}$$

where $\mathbf{v}_i$ is the velocity of link i, $\boldsymbol{\Phi}_i$ is the motion matrix of joint i, and $\dot{\boldsymbol{q}}_i$ is the vector of joint velocity variables for joint i.

The equivalent formula for accelerations is obtained by differentiating (3.71), giving

$$\mathbf{a}_i = \mathbf{a}_{p(i)} + \boldsymbol{\Phi}_i \ddot{\boldsymbol{q}}_i + \dot{\boldsymbol{\Phi}}_i \dot{\boldsymbol{q}}_i \,, \quad (\mathbf{a}_0 = \mathbf{0}) \,, \tag{3.72}$$

where $\mathbf{a}_i$ is the acceleration of link i, and $\ddot{\boldsymbol{q}}_i$ is the vector of joint acceleration variables.

The effect of a uniform gravitational field on the mechanism can be simulated by initializing $\mathbf{a}_0$ to $-\mathbf{a}_g$ instead of zero, where $\mathbf{a}_g$ is the gravitational acceleration vector. In this case, $\mathbf{a}_i$ is not the true acceleration of link i, but the sum of its true acceleration and $-\mathbf{a}_g$.

Step 2

Calculate the equation of motion for each link. This step computes the forces required to cause the accelerations calculated in step 1. The equation of motion for link i is

$$\mathbf{f}_i^{\mathrm{a}} = \boldsymbol{I}_i \mathbf{a}_i + \mathbf{v}_i \times \boldsymbol{I}_i \mathbf{v}_i \,, \tag{3.73}$$

where $\boldsymbol{I}_i$ is the spatial inertia of link i, and $\mathbf{f}_i^{\mathrm{a}}$ is the net force acting on link i.

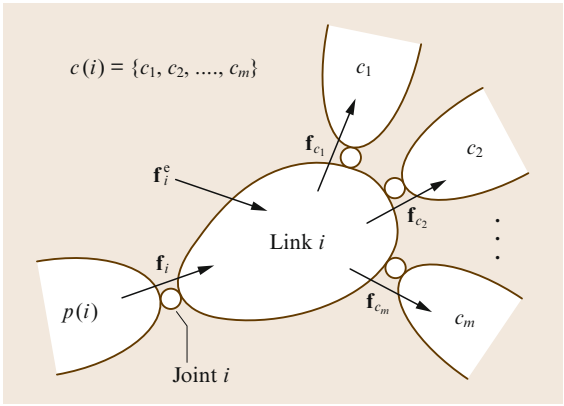


Fig. 3.6 Forces acting on link i

Step 3

Calculate the spatial force across each joint. Referring to Fig. 3.6, the net force acting on link i is

$$\mathbf{f}_i^{\mathrm{a}} = \mathbf{f}_i^{\mathrm{e}} + \mathbf{f}_i - \sum_{j \in c(i)} \mathbf{f}_j \,,$$

where $\mathbf{f}_i$ is the force transmitted across joint i, $\mathbf{f}_i^{\mathrm{e}}$ is the sum of all relevant external forces acting on link i, and $c(i)$ is the set of children of link i. Rearranging this equation gives the following recursive formula for calculating the joint forces

$$\mathbf{f}_i = \mathbf{f}_i^{\mathrm{a}} - \mathbf{f}_i^{\mathrm{e}} + \sum_{j \in c(i)} \mathbf{f}_j \,, \tag{3.74}$$

where i iterates from N_B to 1.

$\mathbf{f}_i^{\mathrm{e}}$ may include contributions from springs, dampers, force fields, contact with the environment, and so on, but its value is assumed to be known, or at least to be calculable from known quantities.

If gravity has not been simulated by a fictitious base acceleration, then the gravitational force acting on link i must be included in $\mathbf{f}_i^{\mathrm{e}}$.

Step 4

Calculate the joint force variables, $\boldsymbol{\tau}_i$. By definition, they are given by the equation

$$\boldsymbol{\tau}_i = \boldsymbol{\Phi}_i^{\mathrm{T}} \mathbf{f}_i \,. \tag{3.75}$$

Coordinate-Free Algorithm

Equations (3.71)–(3.75) imply the algorithm shown in Algorithm 3.1, which is the coordinate-free version of the RNEA. This is the simplest form of the algorithm, and it is suitable for mathematical analysis and related purposes. However, it is not suitable for numerical

computation because a numerical version must use coordinate vectors.

Algorithm 3.1 Coordinate-free recursive Newton–Euler algorithm (RNEA) for inverse dynamics

$\mathbf{v}_0 = \mathbf{0}$
$\mathbf{a}_0 = -\mathbf{a}_g$
for $i = 1$ **to** N_B **do**
  $\mathbf{v}_i = \mathbf{v}_{p(i)} + \boldsymbol{\Phi}_i \dot{\boldsymbol{q}}_i$
  $\mathbf{a}_i = \mathbf{a}_{p(i)} + \boldsymbol{\Phi}_i \ddot{\boldsymbol{q}}_i + \dot{\boldsymbol{\Phi}}_i \dot{\boldsymbol{q}}_i$
  $\mathbf{f}_i = \boldsymbol{I}_i \mathbf{a}_i + \mathbf{v}_i \times \boldsymbol{I}_i \mathbf{v}_i - \mathbf{f}_i^e$
end for
for $i = N_B$ **to** 1 **do**
  $\boldsymbol{\tau}_i = \boldsymbol{\Phi}_i^T \mathbf{f}_i$
  if $p(i) \neq 0$ **then**
    $\mathbf{f}_{p(i)} = \mathbf{f}_{p(i)} + \mathbf{f}_i$
  end if
end for

Link-Coordinates Algorithm

In general, we say that an algorithm is implemented in link coordinates if a coordinate system is defined for each link, and the calculations pertaining to link i are performed in the coordinate system associated with link i. The alternative is to implement the algorithm in absolute coordinates, in which case all calculations are performed in a single coordinate system, typically that of the base link. In practice, the RNEA is more computationally efficient when implemented in link coordinates, and the same is true of most other dynamics algorithms.

To convert the RNEA to link coordinates, we first examine the equations to see which ones involve quantities from more than one link. Equations (3.73) and (3.75) each involve quantities pertaining to link i only, and therefore need no modification. Such equations are said to be local to link i. The remaining equations involve quantities from more than one link, and therefore require the insertion of coordinate transformation matrices. The modified versions of (3.71), (3.72), and (3.74) are

$$\mathbf{v}_i = {}^i\boldsymbol{X}_{p(i)}\, \mathbf{v}_{p(i)} + \boldsymbol{\Phi}_i \dot{\boldsymbol{q}}_i \,, \tag{3.76}$$

$$\mathbf{a}_i = {}^i\boldsymbol{X}_{p(i)}\, \mathbf{a}_{p(i)} + \boldsymbol{\Phi}_i \ddot{\boldsymbol{q}}_i + \dot{\boldsymbol{\Phi}}_i \dot{\boldsymbol{q}}_i \,, \tag{3.77}$$

and

$$\mathbf{f}_i = \mathbf{f}_i^a - {}^i\boldsymbol{X}_0^F\, {}^0\mathbf{f}_i^e + \sum_{j \in c(i)} {}^i\boldsymbol{X}_j^F\, \mathbf{f}_j \,. \tag{3.78}$$

Equation (3.78) assumes that external forces are expressed in absolute (i. e., link 0) coordinates.

Algorithm 3.2 Recursive Newton–Euler algorithm using spatial vectors

inputs: $\mathbf{q}$, $\dot{\mathbf{q}}$, $\ddot{\mathbf{q}}$, *model* , ${}^0\mathbf{f}_i^e$
output: $\boldsymbol{\tau}$
model data : N_B, jtype(i), $p(i)$, $\boldsymbol{X}_L(i)$, $\boldsymbol{I}_i$

$\mathbf{v}_0 = \mathbf{0}$
$\mathbf{a}_0 = -\mathbf{a}_g$
for $i = 1$ **to** N_B **do**
  $\boldsymbol{X}_J(i) = \text{xjcalc}(\text{jtype}(i), \boldsymbol{q}_i)$
  ${}^i\boldsymbol{X}_{p(i)} = \boldsymbol{X}_J(i)\, \boldsymbol{X}_L(i)$
  if $p(i) \neq 0$ **then**
    ${}^i\boldsymbol{X}_0 = {}^i\boldsymbol{X}_{p(i)}\, {}^{p(i)}\boldsymbol{X}_0$
  end if
  $\boldsymbol{\Phi}_i = \text{pcalc}(\text{jtype}(i), \boldsymbol{q}_i)$
  $\mathring{\boldsymbol{\Phi}}_i = \text{pdcalc}(\text{jtype}(i), \boldsymbol{q}_i, \dot{\boldsymbol{q}}_i)$
  $\mathbf{v}_i = {}^i\boldsymbol{X}_{p(i)}\, \mathbf{v}_{p(i)} + \boldsymbol{\Phi}_i \dot{\boldsymbol{q}}_i$
  $\boldsymbol{\zeta}_i = \mathring{\boldsymbol{\Phi}}_i \dot{\boldsymbol{q}}_i + \mathbf{v}_i \times \boldsymbol{\Phi}_i \dot{\boldsymbol{q}}_i$
  $\mathbf{a}_i = {}^i\boldsymbol{X}_{p(i)}\, \mathbf{a}_{p(i)} + \boldsymbol{\Phi}_i \ddot{\boldsymbol{q}}_i + \boldsymbol{\zeta}_i$
  $\mathbf{f}_i = \boldsymbol{I}_i \mathbf{a}_i + \mathbf{v}_i \times \boldsymbol{I}_i \mathbf{v}_i - {}^i\boldsymbol{X}_0^{-T}\, {}^0\mathbf{f}_i^e$
end for
for $i = N_B$ **to** 1 **do**
  $\boldsymbol{\tau}_i = \boldsymbol{\Phi}_i^T \mathbf{f}_i$
  if $p(i) \neq 0$ **then**
    $\mathbf{f}_{p(i)} = \mathbf{f}_{p(i)} + {}^i\boldsymbol{X}_{p(i)}^T\, \mathbf{f}_i$
  end if
end for

The complete algorithm is shown in Algorithm 3.2. The function `jtype` returns the type code for joint i; the function `xjcalc` calculates the joint transformation matrix for the specified type of joint; and the functions `pcalc` and `pdcalc` calculate $\boldsymbol{\Phi}_i$ and $\mathring{\boldsymbol{\Phi}}_i$. The formulae used by these functions for a variety of joint types can be found in Tables 2.5 and 2.6, bearing in mind that the rotation matrices that must be used are the transposes of those listed in these tables. In the general case, both `pcalc` and `pdcalc` are needed. However, for most common joint types, $\boldsymbol{\Phi}_i$ is a known constant in link coordinates, and $\mathring{\boldsymbol{\Phi}}_i$ is therefore zero. If it is known in advance that all joints will have this property, then the algorithm can be simplified accordingly. The quantities $\boldsymbol{I}_i$ and $\boldsymbol{X}_L(i)$ are known constants in link coordinates, and are part of the data structure describing the robot mechanism.

The last assignment in the first loop initializes each $\mathbf{f}_i$ to the expression $\mathbf{f}_i^a - {}^i\boldsymbol{X}_0^F\, {}^0\mathbf{f}_i^e$ (using the identity ${}^i\boldsymbol{X}_0^F = {}^i\boldsymbol{X}_0^{-T}$). The summation on the right-hand side of (3.78) is then performed in the second loop. This algorithm includes code to calculate ${}^i\boldsymbol{X}_0$, which is used to transform the external forces to link coordinates. If there are no external forces, then this code can be omitted. If there is only a single external force (e.g., a force

Avidbots Ex1016
Page 100 of 2154

at the end-effector of a robot arm) then this code can be replaced with code that transforms the external force vector successively from one link coordinate system to the next, using ${}^{i}\boldsymbol{X}_{p(i)}$.

Note: although the phrase *link coordinates* suggests that we are using moving coordinate frames, the algorithm is in fact implemented in stationary coordinates that happen to coincide with the moving coordinates at the current instant.

3-D Vector RNEA

The original version of the RNEA was developed and expressed using 3-D vectors (e.g. [3.2, 4]). Algorithm 3.3 shows a special case of this algorithm, in which the joints are assumed to be revolute, and the joint axes are assumed to coincide with the z axes of the link coordinate systems. (Without these assumptions, the equations would be a lot longer.) It also assumes that the external forces are zero.

Algorithm 3.3 Recursive Newton–Euler algorithm in 3-D vectors, for revolute joints only

inputs: $\mathbf{q}$, $\dot{\mathbf{q}}$, $\ddot{\mathbf{q}}$, *model*
output: $\boldsymbol{\tau}$
model data : N_B, $p(i)$, $\boldsymbol{R}_L(i)$, ${}^{p(i)}\boldsymbol{p}_i$, m_i, $\boldsymbol{c}_i$, $\bar{\boldsymbol{I}}_i^{\mathrm{cm}}$

$\boldsymbol{\omega}_0 = \mathbf{0}$
$\dot{\boldsymbol{\omega}}_0 = \mathbf{0}$
$\dot{\boldsymbol{v}}'_0 = -\dot{\boldsymbol{v}}'_{\mathrm{g}}$
for $i = 1$ **to** N_B **do**
  ${}^{i}\boldsymbol{R}_{p(i)} = \mathrm{rotz}(\boldsymbol{q}_i)\,\boldsymbol{R}_L(i)$
  $\boldsymbol{\omega}_i = {}^{i}\boldsymbol{R}_{p(i)}\,\boldsymbol{\omega}_{p(i)} + \hat{\boldsymbol{z}}_i\dot{q}_i$
  $\dot{\boldsymbol{\omega}}_i = {}^{i}\boldsymbol{R}_{p(i)}\,\dot{\boldsymbol{\omega}}_{p(i)} + \left({}^{i}\boldsymbol{R}_{p(i)}\,\boldsymbol{\omega}_{p(i)}\right) \times \hat{\boldsymbol{z}}_i\dot{q}_i + \hat{\boldsymbol{z}}_i\ddot{q}_i$
  $\dot{\boldsymbol{v}}'_i = {}^{i}\boldsymbol{R}_{p(i)} \left(\dot{\boldsymbol{v}}'_{p(i)} + \dot{\boldsymbol{\omega}}_{p(i)} \times {}^{p(i)}\boldsymbol{p}_i + \boldsymbol{\omega}_{p(i)} \times \boldsymbol{\omega}_{p(i)} \times {}^{p(i)}\boldsymbol{p}_i \right)$
  $\boldsymbol{f}_i = m_i(\dot{\boldsymbol{v}}'_i + \dot{\boldsymbol{\omega}}_i \times \boldsymbol{c}_i + \boldsymbol{\omega}_i \times \boldsymbol{\omega}_i \times \boldsymbol{c}_i)$
  $\boldsymbol{n}_i = \bar{\boldsymbol{I}}_i^{\mathrm{cm}}\,\dot{\boldsymbol{\omega}}_i + \boldsymbol{\omega}_i \times \bar{\boldsymbol{I}}_i^{\mathrm{cm}}\,\boldsymbol{\omega}_i + \boldsymbol{c}_i \times \boldsymbol{f}_i$
end for
for $i = N_B$ **to** 1 **do**
  $\tau_i = \hat{\boldsymbol{z}}_i^{\mathrm{T}}\boldsymbol{n}_i$
  if $p(i) \neq 0$ **then**
    $\boldsymbol{f}_{p(i)} = \boldsymbol{f}_{p(i)} + {}^{i}\boldsymbol{R}_{p(i)}^{\mathrm{T}}\boldsymbol{f}_i$
    $\boldsymbol{n}_{p(i)} = \boldsymbol{n}_{p(i)} + {}^{i}\boldsymbol{R}_{p(i)}^{\mathrm{T}}\boldsymbol{n}_i + {}^{p(i)}\boldsymbol{p}_i \times {}^{i}\boldsymbol{R}_{p(i)}^{\mathrm{T}}\boldsymbol{f}_i$
  end if
end for

In this algorithm, $\dot{\boldsymbol{v}}'_{\mathrm{g}}$ is the linear acceleration due to gravity, expressed in base (link 0) coordinates; `rotz` computes the transpose of the matrix shown in (2.2); $\boldsymbol{R}_L(i)$ is the rotational component of $\boldsymbol{X}_L(i)$; ${}^{i}\boldsymbol{R}_{p(i)}$ is the rotational component of ${}^{i}\boldsymbol{X}_{p(i)}$; `pcalc` and `pdcalc` are not used because $\boldsymbol{\Phi}_i$ is the known constant $(\hat{\boldsymbol{z}}^{\mathrm{T}}\ \mathbf{0}^{\mathrm{T}})^{\mathrm{T}}$; $\dot{\boldsymbol{v}}'_i$ is the linear acceleration of the origin of link i coordinates (O_i), and is the linear component of the classical acceleration of link i; ${}^{p(i)}\boldsymbol{p}_i$ is the position of O_i relative to $O_{p(i)}$ expressed in $p(i)$ coordinates; and m_i, $\boldsymbol{c}_i$, and $\bar{\boldsymbol{I}}_i^{\mathrm{cm}}$ are the inertia parameters of link i. (See Table 3.1 for the equations relating these 3-D quantities to the corresponding spatial quantities.)

At first sight, the 3-D vector algorithm looks significantly different from the spatial vector algorithm. Nevertheless, it can be obtained directly from the spatial vector algorithm simply by expanding the spatial vectors to their 3-D components, restricting the joint type to revolute, converting spatial accelerations to classical accelerations (i. e., replacing each instance of $\dot{\boldsymbol{v}}_i$ with $\dot{\boldsymbol{v}}'_i - \boldsymbol{\omega}_i \times \boldsymbol{v}_i$ as per (3.22)), and applying some 3-D vector identities to bring the equations into the form shown in the table. The conversion from spatial to classical acceleration has one interesting side-effect: $\boldsymbol{v}_i$ cancels out of the equation of motion, and therefore does not need to be calculated. As a result, the 3-D version of the algorithm has a slight speed advantage over the spatial version.

3.5.2 The Articulated-Body Algorithm

The ABA is an $O(N_B)$ algorithm for calculating the forward dynamics of a kinematic tree. However, under normal circumstances, $O(N_B) = O(n)$, so we shall refer to it as an $O(n)$ algorithm. The ABA was developed by *Featherstone* [3.1] and is an example of a constraint-propagation algorithm. Given the joint position, velocity, and applied torque/force variables, this algorithm calculates the joint accelerations. With the joint accelerations determined, numerical integration may be used to provide a simulation of the mechanism.

The key concept in the ABA is illustrated in Fig. 3.7. The subtree rooted at link i interacts with the rest of the kinematic tree only through a force $\mathbf{f}_i$ that is transmitted across joint i. Suppose we break the tree at this point, and consider only the motion of the subtree subject to an unknown force, $\mathbf{f}_i$, acting on link i. It is possible to show that the acceleration of link i is related to the applied force according to the equation

$$\mathbf{f}_i = \boldsymbol{I}_i^A\mathbf{a}_i + \mathbf{p}_i^A\ , \tag{3.79}$$

where $\boldsymbol{I}_i^A$ is called the articulated-body inertia of link i in the subtree (which we can now call an articulated body), and $\mathbf{p}_i^A$ is the associated bias force, which is the force required to produce zero acceleration in link i. Note that $\mathbf{p}_i^A$ depends on the velocities of the individual bodies in the articulated body. Equation (3.79) takes into account the complete dynamics of the subtree. Thus, if we happened to know the correct value of $\mathbf{f}_i$, then (3.79)

would immediately give us the correct acceleration of link i.

The reason we are interested in the quantities $\boldsymbol{I}_i^A$ and $\mathbf{p}_i^A$ is that they allow us to calculate $\ddot{\boldsymbol{q}}_i$ from $\mathbf{a}_{p(i)}$, which in turn allows us to calculate $\mathbf{a}_i$, which then allows us to calculate more joint accelerations, and so on. Combining (3.79) with (3.75) and (3.72) gives

$$\boldsymbol{\tau}_i = \boldsymbol{\Phi}_i^{\mathrm{T}} \mathbf{f}_i = \boldsymbol{\Phi}_i^{\mathrm{T}} \left(\boldsymbol{I}_i^A (\mathbf{a}_{p(i)} + \boldsymbol{\Phi}_i \ddot{\boldsymbol{q}}_i + \dot{\boldsymbol{\Phi}}_i \dot{\boldsymbol{q}}_i) + \mathbf{p}_i^A \right) ,$$

which can be solved for $\ddot{\boldsymbol{q}}_i$ to give

$$\ddot{\boldsymbol{q}}_i = \boldsymbol{D}_i \left(\boldsymbol{u}_i - \boldsymbol{U}_i^{\mathrm{T}} \mathbf{a}_{p(i)} \right) , \tag{3.80}$$

where

$$\begin{aligned} \boldsymbol{U}_i &= \boldsymbol{I}_i^A \boldsymbol{\Phi}_i , \\ \boldsymbol{D}_i &= \left(\boldsymbol{\Phi}_i^{\mathrm{T}} \boldsymbol{U}_i \right)^{-1} = \left(\boldsymbol{\Phi}_i^{\mathrm{T}} \boldsymbol{I}_i^A \boldsymbol{\Phi}_i \right)^{-1} , \\ \boldsymbol{u}_i &= \boldsymbol{\tau}_i - \boldsymbol{U}_i^{\mathrm{T}} \boldsymbol{\zeta}_i - \boldsymbol{\Phi}_i^{\mathrm{T}} \mathbf{p}_i^A \end{aligned}$$

and

$$\boldsymbol{\zeta}_i = \dot{\boldsymbol{\Phi}}_i \dot{\boldsymbol{q}}_i = \mathring{\boldsymbol{\Phi}}_i \dot{\boldsymbol{q}}_i + \mathbf{v}_i \times \boldsymbol{\Phi}_i \dot{\boldsymbol{q}}_i .$$

$\mathbf{a}_i$ can then be calculated via (3.72).

It turns out that the articulated-body inertias and bias forces can be calculated efficiently via the recursive formulae

$$\boldsymbol{I}_i^A = \boldsymbol{I}_i + \sum_{j \in c(i)} \left(\boldsymbol{I}_j^A - \boldsymbol{U}_j \boldsymbol{D}_j \boldsymbol{U}_j^{\mathrm{T}} \right) \tag{3.81}$$

and

$$\mathbf{p}_i^A = \mathbf{p}_i + \sum_{j \in c(i)} \left(\mathbf{p}_j^A + \boldsymbol{I}_j^A \boldsymbol{\zeta}_j + \boldsymbol{U}_j \boldsymbol{D}_j \boldsymbol{u}_j \right) , \tag{3.82}$$

where

$$\mathbf{p}_i = \mathbf{v}_i \times \boldsymbol{I}_i \mathbf{v}_i - \mathbf{f}_i^{\mathrm{e}} .$$

These formulae are obtained by examining the relationship between $\mathbf{f}_i$ and $\mathbf{a}_i$ in Fig. 3.7 and assuming that $\boldsymbol{I}_j^A$ and $\mathbf{p}_j^A$ are already known for every $j \in c(i)$. For more details see [3.1, 8, 15, 25].

The complete algorithm is shown in Algorithm 3.4.

Algorithm 3.4 Articulated-body algorithm for forward dynamics

inputs: $\mathbf{q}$, $\dot{\mathbf{q}}$, $\boldsymbol{\tau}$, *model*, ${}^0\mathbf{f}_i^{\mathrm{e}}$
output: $\ddot{\mathbf{q}}$
model data: N_B, jtype(i), $p(i)$, $\boldsymbol{X}_L(i)$, $\boldsymbol{I}_i$

$\mathbf{v}_0 = \mathbf{0}$
$\mathbf{a}_0 = -\mathbf{a}_{\mathrm{g}}$
for $i = 1$ **to** N_B **do**
  $\boldsymbol{X}_J(i) = \mathrm{xjcalc}(\mathrm{jtype}(i), \boldsymbol{q}_i)$
  ${}^i\boldsymbol{X}_{p(i)} = \boldsymbol{X}_J(i) \boldsymbol{X}_L(i)$
  if $p(i) \neq 0$ **then**
    ${}^i\boldsymbol{X}_0 = {}^i\boldsymbol{X}_{p(i)} {}^{p(i)}\boldsymbol{X}_0$
  end if
  $\boldsymbol{\Phi}_i = \mathrm{pcalc}(\mathrm{jtype}(i), \boldsymbol{q}_i)$
  $\mathring{\boldsymbol{\Phi}}_i = \mathrm{pdcalc}(\mathrm{jtype}(i), \boldsymbol{q}_i, \dot{\boldsymbol{q}}_i)$
  $\mathbf{v}_i = {}^i\boldsymbol{X}_{p(i)} \mathbf{v}_{p(i)} + \boldsymbol{\Phi}_i \dot{\boldsymbol{q}}_i$
  $\boldsymbol{\zeta}_i = \mathring{\boldsymbol{\Phi}}_i \dot{\boldsymbol{q}}_i + \mathbf{v}_i \times \boldsymbol{\Phi}_i \dot{\boldsymbol{q}}_i$
  $\boldsymbol{I}_i^A = \boldsymbol{I}_i$
  $\mathbf{p}_i^A = \mathbf{v}_i \times \boldsymbol{I}_i \mathbf{v}_i - {}^i\boldsymbol{X}_0^{-T} {}^0\mathbf{f}_i^{\mathrm{e}}$
end for
for $i = N_B$ **to** 1 **do**
  $\boldsymbol{U}_i = \boldsymbol{I}_i^A \boldsymbol{\Phi}_i$
  $\boldsymbol{D}_i = \left(\boldsymbol{\Phi}_i^{\mathrm{T}} \boldsymbol{U}_i \right)^{-1}$
  $\boldsymbol{u}_i = \boldsymbol{\tau}_i - \boldsymbol{U}_i^{\mathrm{T}} \boldsymbol{\zeta}_i - \boldsymbol{\Phi}_i^{\mathrm{T}} \mathbf{p}_i^A$
  if $p(i) \neq 0$ **then**
    $\boldsymbol{I}_{p(i)}^A = \boldsymbol{I}_{p(i)}^A + {}^i\boldsymbol{X}_{p(i)}^{\mathrm{T}} \left(\boldsymbol{I}_i^A - \boldsymbol{U}_i \boldsymbol{D}_i \boldsymbol{U}_i^{\mathrm{T}} \right) {}^i\boldsymbol{X}_{p(i)}$
    $\mathbf{p}_{p(i)}^A = \mathbf{p}_{p(i)}^A + {}^i\boldsymbol{X}_{p(i)}^{\mathrm{T}} \left(\mathbf{p}_i^A + \boldsymbol{I}_i^A \boldsymbol{\zeta}_i + \boldsymbol{U}_i \boldsymbol{D}_i \boldsymbol{u}_i \right)$
  end if
end for
for $i = 1$ **to** N_B **do**
  $\mathbf{a}_i = {}^i\boldsymbol{X}_{p(i)} \mathbf{a}_{p(i)}$
  $\ddot{\boldsymbol{q}}_i = \boldsymbol{D}_i \left(\boldsymbol{u}_i - \boldsymbol{U}_i^{\mathrm{T}} \mathbf{a}_i \right)$
  $\mathbf{a}_i = \mathbf{a}_i + \boldsymbol{\Phi}_i \ddot{\boldsymbol{q}}_i + \boldsymbol{\zeta}_i$
end for

It is expressed in link coordinates, as per the RNEA in Algorithm 3.2. It makes a total of three passes through the kinematic tree. The first pass iterates from the base out to the tips; it calculates the link velocities using (3.76), the velocity-product term $\boldsymbol{\zeta}_i = \dot{\boldsymbol{\Phi}}_i \dot{\boldsymbol{q}}_i$, and it initializes the variables $\boldsymbol{I}_i^A$ and $\mathbf{p}_i^A$ to the values $\boldsymbol{I}_i$ and $\mathbf{p}_i (= \mathbf{v}_i \times \boldsymbol{I}_i \mathbf{v}_i - {}^i\boldsymbol{X}_0^F {}^0\mathbf{f}_i^{\mathrm{e}})$, respectively. The second pass iterates from the tips back to the base; it calculates the articulated-body inertia and bias force for each link using (3.81) and (3.82). The third pass iterates from the base to the tips; it calculates the link and joint accelerations using (3.80) and (3.77).

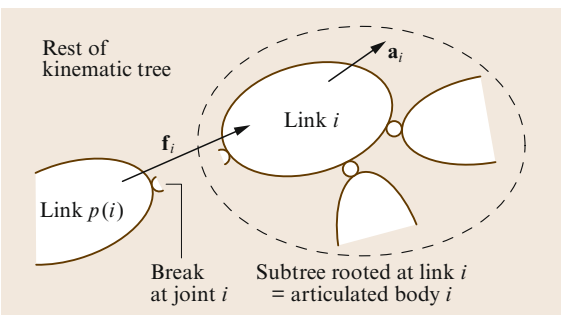


Fig. 3.7 Definition of articulated body i

3.5.3 The Composite-Rigid-Body Algorithm

The CRBA is an algorithm for calculating the joint-space inertia matrix (JSIM) of a kinematic tree. The most common use for the CRBA is as part of a forward dynamics algorithm. It first appeared as method 3 in [3.5].

Forward dynamics, in joint space, is the task of calculating $\ddot{\mathbf{q}}$ from $\mathbf{q}$, $\dot{\mathbf{q}}$, and $\boldsymbol{\tau}$. Starting from (3.37), the most obvious way to proceed is to calculate $\boldsymbol{H}$ and $\boldsymbol{C}\dot{\mathbf{q}} + \boldsymbol{\tau}_g$, and then solve the linear equation

$$\boldsymbol{H}\ddot{\mathbf{q}} = \boldsymbol{\tau} - (\boldsymbol{C}\dot{\mathbf{q}} + \boldsymbol{\tau}_g) \tag{3.83}$$

for $\ddot{\mathbf{q}}$. If the mechanism is a kinematic tree, then $\boldsymbol{H}$ and $\boldsymbol{C}\dot{\mathbf{q}} + \boldsymbol{\tau}_g$ can be computed in $O(n^2)$ and $O(n)$ operations, respectively, and (3.83) can be solved in $O(n^3)$ operations. Algorithms that take this approach are therefore known collectively as $O(n^3)$ algorithms. However, this figure of $O(n^3)$ should be regarded as the worst-case complexity, since the actual complexity depends on the amount of branching in the tree [3.26]. Furthermore, even in the worst case, the n^3 term has a small coefficient, and does not dominate until approximately $n = 60$.

$\boldsymbol{C}\dot{\mathbf{q}} + \boldsymbol{\tau}_g$ can be calculated using an inverse dynamics algorithm. If $ID(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})$ is the result of an inverse dynamics calculation with arguments $\mathbf{q}$, $\dot{\mathbf{q}}$ and $\ddot{\mathbf{q}}$, then

$$ID(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}) = \boldsymbol{\tau} = \boldsymbol{H}\ddot{\mathbf{q}} + \boldsymbol{C}\dot{\mathbf{q}} + \boldsymbol{\tau}_g \,,$$

so

$$\boldsymbol{C}\dot{\mathbf{q}} + \boldsymbol{\tau}_g = ID(\mathbf{q}, \dot{\mathbf{q}}, \mathbf{0}) \,. \tag{3.84}$$

Thus, the value of $\boldsymbol{C}\dot{\mathbf{q}} + \boldsymbol{\tau}_g$ for a kinematic tree can be calculated efficiently using the RNEA with $\ddot{\mathbf{q}} = \mathbf{0}$.

The key concept in the CRBA is to note that the JSIM only depends on the joint positions, and not their rates. The CRBA makes the simplifying assumption that the rate at each joint is zero. By also assuming that gravity is zero, $\boldsymbol{C}\dot{\mathbf{q}} + \boldsymbol{\tau}_g$ is eliminated from (3.83). Furthermore, for a revolute joint, a unit of joint acceleration applied at the j-th joint produces the j-th column of the JSIM. This partitions the mechanism into two composite rigid bodies connected by the j-th joint, and simplifies the dynamics considerably. This concept has been generalized so that the CRBA may be applied to any joint type within a kinematic tree structure.

It can be shown that the general form of the JSIM for a kinematic tree is

$$\boldsymbol{H}_{ij} = \begin{cases} \boldsymbol{\Phi}_i^{\mathrm{T}} \boldsymbol{I}_i^{\mathrm{C}} \boldsymbol{\Phi}_j & \text{if } i \in c^*(j) \\ \boldsymbol{\Phi}_i^{\mathrm{T}} \boldsymbol{I}_j^{\mathrm{C}} \boldsymbol{\Phi}_j & \text{if } j \in c^*(i) \\ \mathbf{0} & \text{otherwise,} \end{cases} \tag{3.85}$$

where $c^*(i)$ is the set of links in the subtree rooted at link i, including link i itself, and

$$\boldsymbol{I}_i^{\mathrm{C}} = \sum_{j \in c^*(i)} \boldsymbol{I}_j \,. \tag{3.86}$$

See [3.8, 15]. In fact, $\boldsymbol{I}_i^{\mathrm{C}}$ is the inertia of the composite rigid body formed by the rigid assembly of all the links in $c^*(i)$, and this is where the algorithm gets its name.

Equations (3.85) and (3.86) are the basis of the algorithm shown in Algorithm 3.5, which is the CRBA in link coordinates.

Algorithm 3.5 Composite-rigid-body algorithm for calculating the JSIM

inputs: *model* , *RNEA partial results*
output: $\boldsymbol{H}$
model data : $N_B, p(i), \boldsymbol{I}_i$
RNEA data : $\boldsymbol{\Phi}_i, {}^i\boldsymbol{X}_{p(i)}$

$\boldsymbol{H} = \mathbf{0}$
for $i = 1$ **to** N_B **do**
  $\boldsymbol{I}_i^{\mathrm{C}} = \boldsymbol{I}_i$
end for
for $i = N_B$ **to** 1 **do**
  $\boldsymbol{F} = \boldsymbol{I}_i^{\mathrm{C}} \boldsymbol{\Phi}_i$
  $\boldsymbol{H}_{ii} = \boldsymbol{\Phi}_i^{\mathrm{T}} \boldsymbol{F}$
  if $p(i) \neq 0$ **then**
    $\boldsymbol{I}_{p(i)}^{\mathrm{C}} = \boldsymbol{I}_{p(i)}^{\mathrm{C}} + {}^i\boldsymbol{X}_{p(i)}^{\mathrm{T}} \boldsymbol{I}_i^{\mathrm{C}} {}^i\boldsymbol{X}_{p(i)}$
  end if
  $j = i$
  while $p(j) \neq 0$ **do**
    $\boldsymbol{F} = {}^j\boldsymbol{X}_{p(j)}^{\mathrm{T}} \boldsymbol{F}$
    $j = p(j)$
    $\boldsymbol{H}_{ij} = \boldsymbol{F}^{\mathrm{T}} \boldsymbol{\Phi}_j$
    $\boldsymbol{H}_{ji} = \boldsymbol{H}_{ij}^{\mathrm{T}}$
  end while
end for

This algorithm assumes that the matrices ${}^i\boldsymbol{X}_{p(i)}$ and $\boldsymbol{\Phi}_i$ have already been calculated, e.g., during the calculation of $\boldsymbol{C}\dot{\mathbf{q}} + \boldsymbol{\tau}_g$. If this is not the case, then the relevant lines from Algorithm 3.2 can be inserted into the first loop. The matrix $\boldsymbol{F}$ is a local variable. The first step, $\boldsymbol{H} = \mathbf{0}$, can be omitted if there are no branches in the tree.

Having calculated $\boldsymbol{C}\dot{\mathbf{q}} + \boldsymbol{\tau}_g$ and $\boldsymbol{H}$, the final step is to solve (3.83) for $\ddot{\mathbf{q}}$. This can be done using a standard Cholesky or $\boldsymbol{LDL}^{\mathrm{T}}$ factorization. Note that $\boldsymbol{H}$ can be highly ill-conditioned [3.27], reflecting an underlying ill-conditioning of the kinematic tree itself, so it is recommended to use double-precision arithmetic for every step in the forward dynamics calculation. (This advice applies also to the ABA.)

Exploiting Sparsity

Equation (3.85) implies that some elements of $\boldsymbol{H}$ will automatically be zero if there are branches in the kinematic tree. An example of this effect is shown in Fig. 3.8. Observe that nearly half of the elements are zero. It is possible to exploit this sparsity using the factorization algorithms described in [3.26]. Depending on the amount of branching in the tree, the sparse algorithms can run many times faster than the standard algorithms.

3.5.4 Operational-Space Inertia Matrix

Two different algorithms are presented to calculate the OSIM. The first is an $O(n^3)$ algorithm that uses the basic definition of the OSIM along with efficient factorization of the JSIM. The second is an $O(n)$ algorithm that is based on efficient solution of the forward dynamics problem.

Algorithm Using Basic Definition

If a robot has a relatively small number of freedoms (e.g., six), then the most efficient method for calculating the OSIM is via (3.52). The procedure is as follows:

1. Calculate $\boldsymbol{H}$ via the CRBA.
2. Factorize $\boldsymbol{H}$ into $\boldsymbol{H} = \boldsymbol{L}\boldsymbol{L}^{\mathrm{T}}$ (Cholesky factorization).
3. Use back-substitution to calculate $\boldsymbol{Y} = \boldsymbol{L}^{-1}\boldsymbol{J}^{\mathrm{T}}$.
4. $\boldsymbol{\Lambda}^{-1} = \boldsymbol{Y}^{\mathrm{T}}\boldsymbol{Y}$.
5. Factorize $\boldsymbol{\Lambda}^{-1}$ (optional).

The final step is only possible if the end-effector has a full six DOFs, and is only necessary if the application requires $\boldsymbol{\Lambda}$ rather than $\boldsymbol{\Lambda}^{-1}$. In the second step, an $\boldsymbol{L}\boldsymbol{D}\boldsymbol{L}^{\mathrm{T}}$ factorization can be used instead of Cholesky, or one can use one of the efficient factorizations described in [3.26] for branched kinematic trees.

The other terms in (3.38) can be calculated via (3.53) and (3.54). In particular, (3.38) can be rewritten in the form

$$\dot{\mathbf{v}} + \boldsymbol{\Lambda}^{-1}(\boldsymbol{x})[\boldsymbol{\mu}(\boldsymbol{x}, \mathbf{v}) + \boldsymbol{\rho}(\boldsymbol{x})] = \boldsymbol{\Lambda}^{-1}(\boldsymbol{x})\,\mathbf{f}\,, \qquad (3.87)$$

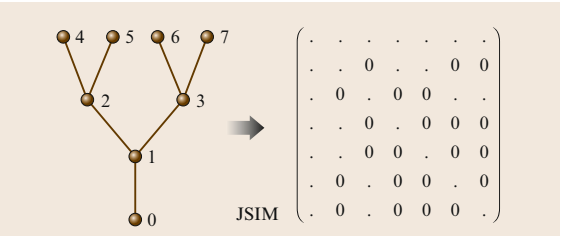


Fig. 3.8 Branch-induced sparsity: branches in the kinematic tree cause certain elements in the JSIM to be zero

and the quantity $\boldsymbol{\Lambda}^{-1}(\boldsymbol{\mu} + \boldsymbol{\rho})$ can be calculated from the formula

$$\boldsymbol{\Lambda}^{-1}(\boldsymbol{\mu} + \boldsymbol{\rho}) = \boldsymbol{J}\boldsymbol{H}^{-1}(\boldsymbol{C}\dot{\mathbf{q}} + \boldsymbol{\tau}_{\mathrm{g}}) - \dot{\boldsymbol{J}}\dot{\mathbf{q}}\,. \qquad (3.88)$$

The term $\dot{\boldsymbol{J}}\dot{\mathbf{q}}$ is the velocity-product acceleration of the end-effector (3.50). It is calculated as a by-product of calculating $\boldsymbol{C}\dot{\mathbf{q}} + \boldsymbol{\tau}_{\mathrm{g}}$ via the RNEA (3.84). Specifically, $\dot{\boldsymbol{J}}\dot{\mathbf{q}} = \mathbf{a}_{\mathrm{ee}} - \mathbf{a}_0$, where $\mathbf{a}_{\mathrm{ee}}$ is the calculated acceleration of the end-effector (expressed in the same coordinates as $\dot{\mathbf{v}}$) and $\mathbf{a}_0$ is the acceleration of the base ($-\mathbf{a}_{\mathrm{g}}$).

$O(n)$ Algorithm

For a sufficiently large value of n, it becomes more efficient to use an $O(n)$ algorithm. Several such algorithms can be found in [3.28–30]. In this section, a more straightforward algorithm is given, which is based on an $O(n)$ calculation of the joint-space forward dynamics problem, e.g., via the ABA. It is a variation of the unit force method [3.29] and computes the inverse of the OSIM.

Starting with (3.87), observe that $\boldsymbol{\Lambda}^{-1}$ is a function of position only, and certain terms in the dynamic equations can be neglected without affecting its value. Specifically, if the joint rates, $\dot{\mathbf{q}}$, joint forces, $\boldsymbol{\tau}$, and gravitational forces are all set to zero, the value of $\boldsymbol{\Lambda}$ will remain unchanged. Under these conditions,

$$\dot{\mathbf{v}} = \boldsymbol{\Lambda}^{-1}\mathbf{f}\,. \qquad (3.89)$$

Let us define $\hat{\mathbf{e}}_i$ to be a 6-D coordinate vector with a 1 in the i-th coordinate and zeros elsewhere. If we set $\mathbf{f} = \hat{\mathbf{e}}_i$ in (3.89), then $\dot{\mathbf{v}}$ will equal column i of $\boldsymbol{\Lambda}^{-1}$. Let us also define the function $FD(i, j, \mathbf{q}, \dot{\mathbf{q}}, \mathbf{a}_0, \boldsymbol{\tau}, \mathbf{f})$, which performs a forward-dynamics calculation and returns the true acceleration of link i (i. e., $\mathbf{a}_i - \mathbf{a}_0$), expressed in the same coordinates as $\mathbf{f}$ (typically the base coordinates). The arguments $\mathbf{q}$, $\dot{\mathbf{q}}$ and $\boldsymbol{\tau}$ set the values of the joint position, velocity, and force variables, while j and $\mathbf{f}$ specify that an external force of $\mathbf{f}$ is to be applied to link j. The argument $\mathbf{a}_0$ specifies a fictitious base acceleration to include gravitational effects, and is set to either $\mathbf{0}$ or $-\mathbf{a}_{\mathrm{g}}$.

With these definitions, we have

$$(\boldsymbol{\Lambda}^{-1})^i = FD(ee, ee, \mathbf{q}, \mathbf{0}, \mathbf{0}, \mathbf{0}, \hat{\mathbf{e}}_i) \qquad (3.90)$$

and

$$\boldsymbol{\Lambda}^{-1}(\boldsymbol{\mu} + \boldsymbol{\rho}) = -FD(ee, ee, \mathbf{q}, \dot{\mathbf{q}}, -\mathbf{a}_{\mathrm{g}}, \boldsymbol{\tau}, \mathbf{0})\,, \qquad (3.91)$$

where $(\boldsymbol{\Lambda}^{-1})^i$ is column i of $\boldsymbol{\Lambda}^{-1}$, and ee is the body number of the end-effector. It therefore follows that the coefficients of (3.87) can be calculated using the algorithm shown in Algorithm 3.6. This algorithm is $O(n)$.

Algorithm 3.6 Algorithm to compute the inverse of the operational-space inertia matrix and other terms

for $j = 1$ **to** 6 **do**
  $\dot{\mathbf{v}}^j = FD(ee, ee, \mathbf{q}, \mathbf{0}, \mathbf{0}, \mathbf{0}, \hat{\mathbf{e}}_j)$
end for
$\boldsymbol{\Lambda}^{-1} = \left[\dot{\mathbf{v}}^1\ \dot{\mathbf{v}}^2\ \cdots\ \dot{\mathbf{v}}^6\right]$
$\boldsymbol{\Lambda}^{-1}(\boldsymbol{\mu} + \boldsymbol{\rho}) = -FD(ee, ee, \mathbf{q}, \dot{\mathbf{q}}, -\mathbf{a}_g, \boldsymbol{\tau}, \mathbf{0})$

The efficiency of the algorithm may be increased significantly when computing $\boldsymbol{\Lambda}^{-1}$ by noting that: (1) $\mathbf{v}_i$, $\boldsymbol{\zeta}_i$, and $\boldsymbol{\tau}_i$ in the ABA calculation (Algorithm 3.4) may be set to zero, and (2) $\boldsymbol{I}_i^A$ and the quantities that depend upon it ($\boldsymbol{U}_i$ and $\boldsymbol{D}_i$) need only be computed once, since they do not vary with the applied force. Also, note that the algorithm may be applied to multiple end-effectors by modifying *FD* to accept a list of end-effector body numbers in its first argument, and return a composite vector containing the accelerations of all the specified bodies. Algorithm 3.6 is then enclosed in a `for` loop that controls the second argument to *FD*, and iterates over all of the end-effector body numbers [3.20].

However, for the case of branched mechanisms with multiple end-effectors, several published algorithms achieve even better efficiency, and should be used instead [3.20, 29, 31–33]. The best achievable complexity is $O(n + md + m^2)$, where m is the number of end-effectors and d is the depth of the system's connectivity tree [3.33]; but the fastest algorithm for a typical humanoid exploits branch-induced sparsity [3.32, 33].

3.6 Kinematic Loops

All of the algorithms in the last section were for kinematic trees. In this section, a final algorithm is provided for the forward dynamics of closed-loop systems. The algorithm supplements the dynamic equations of motion for a spanning tree of the closed-loop system with the loop-closure constraint equations. Three different methods are given to solve the resulting linear system of equations. An efficient algorithm is given to compute the loop-closure constraints.

Systems with closed kinematic loops exhibit more complicated dynamics than kinematic trees. For example:

1. The degree of motion freedom of a kinematic tree is fixed, but that of a closed-loop system can vary.
2. The degree of instantaneous motion freedom is always the same as the degree of finite motion freedom in a kinematic tree, but they can be different in a closed-loop system.
3. Every force in a kinematic tree can be determined, but some forces in a closed-loop system can be indeterminate. This occurs whenever a closed-loop system is overconstrained.

Two examples of these phenomena are shown in Fig. 3.9. The mechanism in Fig. 3.9a has no finite motion freedom, but it has two degrees of infinitesimal motion freedom. The mechanism in Fig. 3.9b has one degree of freedom when $\theta \neq 0$, but if $\theta = 0$ then the two arms, A and B, are able to move independently, and the mechanism has two degrees of freedom. Moreover, at the boundary between these two motion regimes, the mechanism has three degrees of infinitesimal motion freedom. Both these mechanisms are planar, and are therefore overconstrained. As a result, the out-of-plane components of the joint constraint forces are indeterminate. This kind of indeterminacy has no effect on the motions of these mechanisms, but it does complicate the calculation of their dynamics.

3.6.1 Formulation of Closed-Loop Algorithm

A closed-loop system can be modeled as a spanning tree subject to a set of loop-closure constraint forces. If

$$\boldsymbol{H}\ddot{\mathbf{q}} + \boldsymbol{C}\dot{\mathbf{q}} + \boldsymbol{\tau}_g = \boldsymbol{\tau}$$

is the equation of motion of the spanning tree on its own, then the equation of motion for the closed-loop system is

$$\boldsymbol{H}\ddot{\mathbf{q}} + \boldsymbol{C}\dot{\mathbf{q}} + \boldsymbol{\tau}_g = \boldsymbol{\tau} + \boldsymbol{\tau}^a + \boldsymbol{\tau}^c\,, \tag{3.92}$$

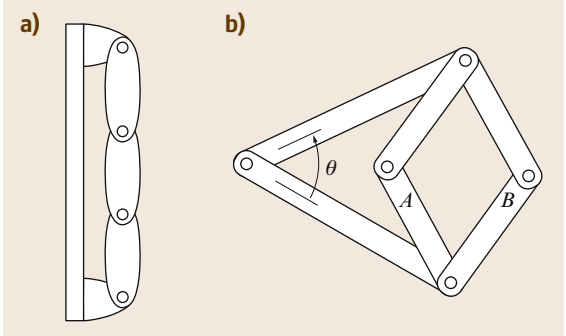


Fig.3.9a,b Pathological closed-loop systems. **(a)** Example of varying motion freedom, **(b)** Example with differing finite and infinitesimal motion freedoms

where $\boldsymbol{\tau}^{\mathrm{a}}$ and $\boldsymbol{\tau}^{\mathrm{c}}$ are vectors of loop-closure active and constraint forces, respectively, expressed in the generalized force coordinates of the spanning tree. $\boldsymbol{\tau}^{\mathrm{a}}$ is a known quantity, and $\boldsymbol{\tau}^{\mathrm{c}}$ is unknown. $\boldsymbol{\tau}^{\mathrm{a}}$ comes from the force elements acting at the loop-closing joints (springs, dampers and actuators). If there are no such force elements, then $\boldsymbol{\tau}^{\mathrm{a}} = \mathbf{0}$.

The loop-closure constraints restrict the motion of the spanning tree. At the acceleration level, these constraints can be expressed in the form of a linear equation,

$$\boldsymbol{L}\ddot{\mathbf{q}} = \mathbf{l}\,, \tag{3.93}$$

where $\boldsymbol{L}$ is an $n^{\mathrm{c}} \times n$ matrix. n^{c} is the number of constraints due to the loop-closing joints, and is given by the formula

$$n^{\mathrm{c}} = \sum_{k=N_B+1}^{N_J} n_k^{\mathrm{c}}\,, \tag{3.94}$$

where n_k^{c} is the number of constraints imposed by joint k. If $\mathrm{rank}(\boldsymbol{L}) < n^{\mathrm{c}}$ then the loop-closure constraints are linearly dependent, and the closed-loop mechanism is overconstrained. The mobility of a closed-loop system (i. e., its degree of motion freedom) is given by the formula

$$\text{mobility} = n - \mathrm{rank}(\boldsymbol{L})\,. \tag{3.95}$$

Given a constraint equation in the form of (3.93), it follows that the constraint forces can be expressed in the form

$$\boldsymbol{\tau}^{\mathrm{c}} = \boldsymbol{L}^{\mathrm{T}}\boldsymbol{\lambda}\,, \tag{3.96}$$

where $\boldsymbol{\lambda} = \left(\boldsymbol{\lambda}_{N_B+1}^{\mathrm{T}} \cdots \boldsymbol{\lambda}_{N_J}^{\mathrm{T}}\right)^{\mathrm{T}}$ is an $n^{\mathrm{c}} \times 1$ vector of unknown constraint-force variables (or Lagrange multipliers). If the mechanism is overconstrained, then $\boldsymbol{L}^{\mathrm{T}}$ will have a null space, and the component of $\boldsymbol{\lambda}$ lying in this null space will be indeterminate.

It is often possible to identify redundant constraints in advance. For example, if a kinematic loop is known to be planar, then the out-of-plane loop-closure constraints are redundant. In these circumstances, it is advantageous to remove the corresponding rows of $\boldsymbol{L}$ and elements of $\mathbf{l}$ and $\boldsymbol{\lambda}$. The removed elements of $\boldsymbol{\lambda}$ can be assigned a value of zero.

Combining (3.92), (3.93), and (3.96) produces the following equation of motion for a closed-loop system

$$\begin{pmatrix} \boldsymbol{H} & \boldsymbol{L}^{\mathrm{T}} \\ \boldsymbol{L} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \ddot{\mathbf{q}} \\ -\boldsymbol{\lambda} \end{pmatrix} = \begin{pmatrix} \boldsymbol{\tau} + \boldsymbol{\tau}^{\mathrm{a}} - (\boldsymbol{C}\dot{\mathbf{q}} + \boldsymbol{\tau}_{\mathrm{g}}) \\ \mathbf{l} \end{pmatrix}. \tag{3.97}$$

The system matrix is symmetric, but indefinite. If $\boldsymbol{L}$ has full rank, then the system matrix will be nonsingular, otherwise it will be singular, and one or more elements of $\boldsymbol{\lambda}$ will be indeterminate.

Equation (3.97) can be solved in any of the following ways:

1. Solve it directly for $\ddot{\mathbf{q}}$ and $\boldsymbol{\lambda}$.
2. Solve for $\boldsymbol{\lambda}$ first, and then use the result to solve for $\ddot{\mathbf{q}}$.
3. Solve (3.93) for $\ddot{\mathbf{q}}$, substitute the result into (3.92), eliminate the unknown constraint forces, and solve for the remaining unknowns.

Method 1 is the simplest, but generally also the least efficient. This method is appropriate when the system matrix is nonsingular. As the size of the system matrix is $(n + n^{\mathrm{c}}) \times (n + n^{\mathrm{c}})$, this method is $O((n + n^{\mathrm{c}})^3)$.

Method 2 is particularly useful if $n \gg n^{\mathrm{c}}$, and offers the opportunity to use $O(n)$ algorithms on the spanning tree [3.6]. From (3.97),

$$\boldsymbol{L}\boldsymbol{H}^{-1}\boldsymbol{L}^{\mathrm{T}}\boldsymbol{\lambda} = \mathbf{l} - \boldsymbol{L}\boldsymbol{H}^{-1}\left[\boldsymbol{\tau} + \boldsymbol{\tau}^{\mathrm{a}} - (\boldsymbol{C}\dot{\mathbf{q}} + \boldsymbol{\tau}_{\mathrm{g}})\right]\,. \tag{3.98}$$

This equation can be formulated in $O(n\,(n^{\mathrm{c}})^2)$ operations via $O(n)$ algorithms, and solved in $O((n^{\mathrm{c}})^3)$. Once $\boldsymbol{\lambda}$ is known, $\boldsymbol{\tau}^{\mathrm{c}}$ can be calculated via (3.96) in $O(n\,n^{\mathrm{c}})$ operations, and (3.92) solved by an $O(n)$ algorithm; so the total complexity is $O(n\,(n^{\mathrm{c}})^2 + (n^{\mathrm{c}})^3)$. If $\boldsymbol{L}$ is rank deficient, then $\boldsymbol{L}\boldsymbol{H}^{-1}\boldsymbol{L}^{\mathrm{T}}$ will be singular; but it is still a positive-semidefinite matrix, and presents a slightly easier factorization problem than a singular instance of the indefinite system matrix in (3.97).

Method 3 is useful if $n - n^{\mathrm{c}}$ is small, or if $\boldsymbol{L}$ is expected to be rank deficient. Equation (3.93) is solved using a special version of Gaussian elimination (or similar procedure), which is equipped with a numerical rank test, and which is designed to solve underdetermined systems. The solution is an equation of the form

$$\ddot{\mathbf{q}} = \boldsymbol{K}\boldsymbol{y} + \ddot{\mathbf{q}}_0\,,$$

where $\ddot{\mathbf{q}}_0$ is any particular solution to (3.93), $\boldsymbol{K}$ is an $n \times (n - \mathrm{rank}(\boldsymbol{L}))$ matrix with the property $\boldsymbol{L}\boldsymbol{K} = \mathbf{0}$, and $\boldsymbol{y}$ is a vector of $n - \mathrm{rank}(\boldsymbol{L})$ unknowns. (Typically, $\boldsymbol{y}$ is a linearly independent subset of the elements of $\ddot{\mathbf{q}}$.) Substituting this expression for $\ddot{\mathbf{q}}$ into (3.92), and premultiplying both sides by $\boldsymbol{K}^{\mathrm{T}}$ to eliminate $\boldsymbol{\tau}^{\mathrm{c}}$, produces

$$\boldsymbol{K}^{\mathrm{T}}\boldsymbol{H}\,\boldsymbol{K}\,\boldsymbol{y} = \boldsymbol{K}^{\mathrm{T}}\left(\boldsymbol{\tau} + \boldsymbol{\tau}^{\mathrm{a}} - (\boldsymbol{C}\dot{\mathbf{q}} + \boldsymbol{\tau}_{\mathrm{g}}) - \boldsymbol{H}\ddot{\mathbf{q}}_0\right)\,. \tag{3.99}$$

This method also has cubic complexity, but it can be the most efficient if $n - n^{\mathrm{c}}$ is small. It is also reported to be more stable than method 1 [3.34].

Avidbots Ex1016
Page 106 of 2154

3.6.2 Closed-Loop Algorithm

Algorithms for calculating $\boldsymbol{H}$ and $\boldsymbol{C}\dot{\mathbf{q}}+\boldsymbol{\tau}_{\mathrm{g}}$ can be found in Sect. 3.5.3 and 3.5.1, respectively, which just leaves $\boldsymbol{L}$, $\mathbf{l}$ and $\boldsymbol{\tau}^{\mathrm{a}}$. To keep things simple, we will assume that all loop-closing joints are zero-DOF joints.

There is no loss of generality with this assumption: one simply breaks open the loops by cutting links instead of joints (Fig. 3.10). However, there may be some loss of efficiency. With this assumption, we only need to calculate $\boldsymbol{L}$ and $\mathbf{l}$, since $\boldsymbol{\tau}^{\mathrm{a}}=\mathbf{0}$.

Loop Constraints

In the general case, the velocity constraint equation for loop k is

$$\left(\boldsymbol{\Psi}_k^{\mathrm{c}}\right)^{\mathrm{T}}\left(\mathbf{v}_{s(k)}-\mathbf{v}_{p(k)}\right)=\mathbf{0}\,, \tag{3.100}$$

and the acceleration constraint is

$$\left(\boldsymbol{\Psi}_k^{\mathrm{c}}\right)^{\mathrm{T}}\left(\mathbf{a}_{s(k)}-\mathbf{a}_{p(k)}\right)+\left(\dot{\boldsymbol{\Psi}}_k^{\mathrm{c}}\right)^{\mathrm{T}}\left(\mathbf{v}_{s(k)}-\mathbf{v}_{p(k)}\right)=\mathbf{0}\,. \tag{3.101}$$

However, if every loop-closing joint has zero DOF, then these equations simplify to

$$\mathbf{v}_{s(k)}-\mathbf{v}_{p(k)}=\mathbf{0} \tag{3.102}$$

and

$$\mathbf{a}_{s(k)}-\mathbf{a}_{p(k)}=\mathbf{0}\,. \tag{3.103}$$

Let us define a loop Jacobian, $\boldsymbol{J}_k$, with the property that

$$\mathbf{v}_{s(k)}-\mathbf{v}_{p(k)}=\boldsymbol{J}_k\dot{\mathbf{q}}\,. \tag{3.104}$$

$\boldsymbol{J}_k$ is a $6\times n$ matrix defined by the formula

$$\boldsymbol{J}_k=\left(e_{1k}\boldsymbol{\Phi}_1\;\cdots\;e_{N_Bk}\boldsymbol{\Phi}_{N_B}\right)\,, \tag{3.105}$$

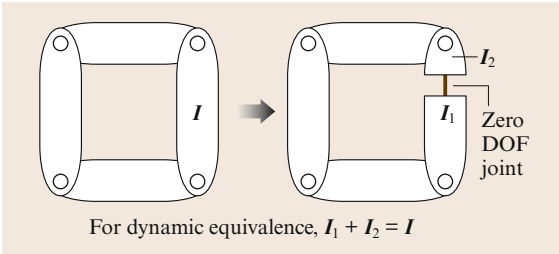


Fig. 3.10 Inserting a zero-DOF joint in preparation for cutting the loop open at that joint

where

$$e_{ik}=\begin{cases}+1 & \text{if } s(k)\in c^*(i) \text{ and } p(k)\notin c^*(i)\,,\\ -1 & \text{if } p(k)\in c^*(i) \text{ and } s(k)\notin c^*(i)\,,\\ 0 & \text{otherwise.}\end{cases}$$

In other words, $e_{ik}=+1$ if joint i lies on the path to $s(k)$ but not the path to $p(k)$; $e_{ik}=-1$ if joint i lies on the path to $p(k)$ but not the path to $s(k)$; and $e_{ik}=0$ if joint i lies on both paths or on neither.

The loop acceleration constraint can now be written

$$\begin{aligned}\mathbf{0}&=\mathbf{a}_{s(k)}-\mathbf{a}_{p(k)}\\&=\boldsymbol{J}_k\ddot{\mathbf{q}}+\dot{\boldsymbol{J}}_k\dot{\mathbf{q}}\\&=\boldsymbol{J}_k\ddot{\mathbf{q}}+\mathbf{a}^{\mathrm{vp}}_{s(k)}-\mathbf{a}^{\mathrm{vp}}_{p(k)}\,,\end{aligned} \tag{3.106}$$

where $\mathbf{a}_i^{\mathrm{vp}}$ is the velocity-product acceleration of link i, which is the acceleration it would have if $\ddot{\mathbf{q}}$ were zero. The velocity-product acceleration of every link is calculated during the calculation of the vector $\boldsymbol{C}\dot{\mathbf{q}}+\boldsymbol{\tau}_{\mathrm{g}}$ (3.84). If the RNEA is used to calculate $\boldsymbol{C}\dot{\mathbf{q}}+\boldsymbol{\tau}_{\mathrm{g}}$, then $\mathbf{a}_i^{\mathrm{vp}}$ will be the value of $\mathbf{a}_i$ calculated by the RNEA with its acceleration argument set to zero.

The matrices $\boldsymbol{L}$ and $\mathbf{l}$ can now be expressed as follows

$$\boldsymbol{L}=\begin{pmatrix}\boldsymbol{L}_{N_B+1}\\ \vdots\\ \boldsymbol{L}_{N_J}\end{pmatrix}\quad\text{and}\quad\mathbf{l}=\begin{pmatrix}\boldsymbol{l}_{N_B+1}\\ \vdots\\ \boldsymbol{l}_{N_J}\end{pmatrix}, \tag{3.107}$$

where

$$\boldsymbol{L}_k=\boldsymbol{J}_k \tag{3.108}$$

and

$$\boldsymbol{l}_k=\mathbf{a}^{\mathrm{vp}}_{p(k)}-\mathbf{a}^{\mathrm{vp}}_{s(k)}\,. \tag{3.109}$$

Constraint Stabilization

In practice, it is necessary to stabilize loop-closure constraints, or they will simply fly apart during simulation because of numerical integration errors. The standard technique is due to *Baumgarte* [3.3, 7, 35], and consists of replacing each constraint equation of the form

$$a_{\mathrm{e}}=0\,,$$

with one of the form

$$a_{\mathrm{e}}+K_{\mathrm{v}}v_{\mathrm{e}}+K_{\mathrm{p}}p_{\mathrm{e}}=0\,,$$

where a_{e}, v_{e}, and p_{e} are the acceleration, velocity, and position errors, respectively, and K_{v} and K_{p} are positive constants. Typically, one chooses a time constant,

t_c, according to how quickly one wants the position and velocity errors to decay. K_p and K_v are then given by the formulae $K_v = 2/t_c$ and $K_p = 1/t_c^2$. However, there is no good rule for choosing t_c. If t_c is too long, then loop-constraint errors accumulate faster than they decay; if t_c is too short, then the equations of motion become excessively stiff, causing a loss of numerical integration accuracy. A reasonable value for a large, slow industrial robot is $t_c = 0.1$, while a smaller, faster robot might need $t_c = 0.01$.

To incorporate stabilization terms into the loop constraint equation, we replace (3.109) with

$$\boldsymbol{l}_k = \mathbf{a}^{\mathrm{vp}}_{p(k)} - \mathbf{a}^{\mathrm{vp}}_{s(k)} - K_v(\mathbf{v}_{s(k)} - \mathbf{v}_{p(k)}) - K_p \mathbf{p}_{ek} \ , \quad (3.110)$$

where $\mathbf{p}_{ek}$ is a vector representing the position error in loop k. In absolute coordinates (i. e., link 0 coordinates), $\mathbf{p}_{ek}$ is given by

$$\mathbf{p}_{ek} = \mathrm{x\_to\_vec}({}^0\boldsymbol{X}_{p(k)}\, \boldsymbol{X}_{L1}^{-1}(k)\, \boldsymbol{X}_{L2}(k)\, {}^{s(k)}\boldsymbol{X}_0) \ , \quad (3.111)$$

where the $\boldsymbol{X}_{L1}(k)$ and $\boldsymbol{X}_{L2}(k)$ transforms are defined in (3.62) and (3.63), and shown in Fig. 3.4 for joint k, and $\mathrm{x\_to\_vec}\left({}^B\boldsymbol{X}_A\right)$ computes a vector approximating the displacement from frame A to frame B, assuming this displacement to be infinitesimal. x_to_vec can be defined as

$$\mathrm{x\_to\_vec}(\boldsymbol{X}) = \frac{1}{2}\begin{pmatrix} X_{23} - X_{32} \\ X_{31} - X_{13} \\ X_{12} - X_{21} \\ X_{53} - X_{62} \\ X_{61} - X_{43} \\ X_{42} - X_{51} \end{pmatrix} . \quad (3.112)$$

Algorithm

Algorithm 3.7 shows an algorithm for calculating $\boldsymbol{L}$ and $\mathbf{l}$ for the special case when all the loop-closing joints have 0-DOF. It combines simplicity with good performance by transforming every quantity that is needed to formulate the loop-closure constraints into a single coordinate system, in this case absolute (link 0) coordinates, so that no further transforms are needed.

The first loop calculates the transforms from absolute to link coordinates, and uses them to transform $\boldsymbol{\Phi}_i$ to absolute coordinates. Only the $\boldsymbol{\Phi}_i$ that are needed in the loop-closure constraints are transformed.

The second loop calculates the nonzero elements of $\boldsymbol{L}$ (which can be sparse), according to (3.105). The inner `while` loop terminates on the root of the loop, which is the highest-numbered common ancestor of links $p(k)$ and $s(k)$. It could be the fixed base if they have no other common ancestor. The second loop ends with the calculation of $\mathbf{l}$, in absolute coordinates, according to (3.110).

Algorithm 3.7 Algorithm to calculate loop-closure constraints

inputs: *model* , *RNEA partial results*
outputs: $\boldsymbol{L}$, $\mathbf{l}$
model data : N_B, $p(i)$, N_J, $p(k)$, $s(k)$, $LB(i)$, $\boldsymbol{X}_{L1}(k)$, $\boldsymbol{X}_{L2}(k)$, K_p, K_v
RNEA data : $\boldsymbol{\Phi}_i$, ${}^i\boldsymbol{X}_{p(i)}$, $\mathbf{v}_{p(k)}$, $\mathbf{v}_{s(k)}$, $\mathbf{a}^{\mathrm{vp}}_{p(k)}$, $\mathbf{a}^{\mathrm{vp}}_{s(k)}$

for $i = 1$ **to** N_B **do**
  if $p(i) \neq 0$ **then**
    ${}^i\boldsymbol{X}_0 = {}^i\boldsymbol{X}_{p(i)}\, {}^{p(i)}\boldsymbol{X}_0$
  end if
  if $LB(i) \neq$ null **then**
    ${}^0\boldsymbol{\Phi}_i = {}^i\boldsymbol{X}_0^{-1}\, \boldsymbol{\Phi}_i$
  end if
end for
$\boldsymbol{L} = \mathbf{0}$
for $k = N_B + 1$ **to** N_J **do**
  $i = p(k)$
  $j = s(k)$
  while $i \neq j$ **do**
    if $i > j$ **then**
      $\boldsymbol{L}_{k,i} = -{}^0\boldsymbol{\Phi}_i$
      $i = p(i)$
    else
      $\boldsymbol{L}_{k,j} = {}^0\boldsymbol{\Phi}_j$
      $j = p(j)$
    end if
  end while
  $\mathbf{a}_e = {}^{s(k)}\boldsymbol{X}_0^{-1}\, \mathbf{a}^{\mathrm{vp}}_{s(k)} - {}^{p(k)}\boldsymbol{X}_0^{-1}\, \mathbf{a}^{\mathrm{vp}}_{p(k)}$
  $\mathbf{v}_e = {}^{s(k)}\boldsymbol{X}_0^{-1}\, \mathbf{v}_{s(k)} - {}^{p(k)}\boldsymbol{X}_0^{-1}\, \mathbf{v}_{p(k)}$
  $\mathbf{p}_e = \mathrm{x\_to\_vec}\left({}^{p(k)}\boldsymbol{X}_0^{-1}\, \boldsymbol{X}_{L1}^{-1}(k)\, \boldsymbol{X}_{L2}(k)\, {}^{s(k)}\boldsymbol{X}_0\right)$
  $\boldsymbol{l}_k = -\mathbf{a}_e - K_v\, \mathbf{v}_e - K_p\, \mathbf{p}_e$
end for

3.7 Conclusions and Further Reading

This chapter has presented the fundamentals of rigid-body dynamics as they apply to robot mechanisms. It has covered the following topics: the spatial vector algebra, which provides a concise notation for describing and implementing dynamics equations and algorithms; the canonical equations of motion that are most fre-

quently used in robotics; how to construct a dynamic model of a robot; and several efficient model-based algorithms for calculating inverse dynamics, forward dynamics, and the joint-space and operational-space inertia matrices.

There are many topics in dynamics that have not been mentioned in this chapter, but can be found in later chapters of this handbook. The dynamics of robots with elastic links and joints is covered in Chap. 11; the problem of identifying the parameters of a dynamic model is covered in Chap. 6; the dynamics of physical contact between a robot and the objects in its environment is described in Chap. 37; and the dynamics of robots with floating bases is described in Chap. 55.

We conclude this chapter by noting that a brief history of robot dynamics can be found in [3.36], and that a more extensive treatment of robot dynamics can be found in books such as [3.8, 10, 15, 29, 37–40]. Finally, some suggestions for further reading are listed below.

3.7.1 Multibody Dynamics

Robot dynamics can be regarded as a subset (or a specific application) of the broader discipline of multibody dynamics. Books on multibody dynamics include [3.3, 14, 35, 41–46]. Of course, multibody dynamics is, in turn, a subset of classical mechanics; and the mathematical foundations of the subject can be found in any good book on classical mechanics, such as [3.13].

3.7.2 Alternative Representations

This chapter has used spatial vectors to express the equations of motion. There are various alternatives to the use of spatial vectors: other kinds of 6-D vector, 3-D vectors, 4×4 matrices, and the spatial operator algebra. All 6-D vector formalisms are similar, but are not exactly the same. The main alternatives to spatial vectors are: screws [3.10–12], motors [3.47], Lie algebras [3.12, 48], and ad hoc notations. (An ad hoc notation is one in which 3-D vectors are grouped into pairs for the purpose of reducing the volume of algebra.) Three-dimensional vectors are the formalism used in most classical mechanics and multibody texts, and are also a precursor to 6-D vector and 4×4 matrix formalisms. 4×4 matrices are popular in robotics because they are very useful for kinematics. However, they are not so useful for dynamics. 4×4 matrix formulations of dynamics can be found in [3.37, 49, 50]. The spatial operator algebra was developed at the Jet Propulsion Laboratory (JPL) by Rodriguez, Jain, and others. It uses $6N$-dimensional vectors and $6N \times 6N$ matrices, the latter regarded as linear operators. Examples of this notation can be found in [3.38, 51–53].

3.7.3 Alternative Formulations

This chapter used a vectorial formulation of the equations of motion that is usually called the Newtonian or Newton–Euler formulation. The main alternative is the Lagrangian formulation, in which the equations of motion are obtained via Lagrange's equation. Examples of the Lagrangian formulation can be found in [3.9, 10, 18, 54, 55]. *Kane*'s method has also been applied in robotics [3.56, 57].

3.7.4 Efficiency

Because of the need for real-time implementation, especially in control, the robotics community has focused on the problem of computational efficiency. For *inverse dynamics*, the $O(n)$ recursive Newton–Euler algorithm (RNEA) of *Luh* et al. [3.4] remains the most important algorithm. Further improvements to the algorithm are given in [3.58, 59]. For *forward dynamics*, the two algorithms presented in this chapter remain the most important for computational considerations: the $O(n)$ articulated-body algorithm (ABA) developed by *Featherstone* [3.1] and the $O(n^3)$ algorithm based on the composite-rigid-body algorithm (CRBA) of *Walker* and *Orin* [3.5]. Improvements were made in the ABA over the years [3.15, 17, 25] so that it was more efficient than the CRBA-based algorithm for decreasingly smaller values of n. However, more recent application of the CRBA to branched kinematic trees [3.26] and robotic systems with motion-controlled appendages [3.60] continue to show the viability of the CRBA approach.

For the *joint-space inertia matrix*, the CRBA [3.5] is the most important algorithm. A number of improvements and modifications have been made over the years to increase its computational efficiency [3.15, 61–63]. For the *operational-space inertia matrix*, efficient $O(n)$ algorithms have been developed [3.28–30] and applied to increasingly complex systems [3.20, 31, 33]. The exploitation of branch-induced sparsity also results in an efficient algorithm [3.32].

3.7.5 Accuracy

Concerns can arise over the numerical accuracy of a dynamics algorithm, the accuracy of a simulation (i. e., numerical integration accuracy), or the accuracy of a dynamic model. The numerical accuracy of dynamics algorithms has received relatively little attention compared with efficiency. The RNEA, CRBA, and ABA have all been tested for accuracy on a large variety of rigid-body systems, but the same cannot be said of most other algorithms. Rigid-body systems are often ill-conditioned, in the sense that a small change

in the applied force (or a model parameter) can produce a large change in the resulting acceleration. This phenomenon was studied by *Featherstone* [3.27], who discovered that the ill-conditioning gets worse with increasing body count, and that it can grow in proportion to $O(n^4)$ in the worst case. Other publications on this topic include [3.8, 34, 64, 65].

3.7.6 Software Packages

A number of software packages have been developed to provide dynamic simulation capabilities for multibody systems, and in particular, robotic systems. Several have been written in MATLAB for ease of integration with other analysis, control, and simulation programs. Many packages are open source, and some are offered at a relatively low cost to the user. They differ in their capabilities in a variety of ways including: speed, topologies and joint models supported, accuracy, underlying dynamic formulation and associated order of complexity, user interface, graphics support, numerical integration routines, integration with other code, application support, and cost. Among those commonly cited are: Adams [3.66], Autolev [3.67], Bullet [3.68], DART [3.69], DynaMechs [3.70], Gazebo [3.71], Open Dynamics Engine [3.72], Robotics Studio [3.73], Robotics Toolbox [3.74], Robotran [3.75, 76], SD/FAST [3.77], Simbody [3.78], SimMechanics [3.79], SYMORO [3.80, 81] and Webots [3.82].

3.7.7 Symbolic Simplification

The technique of symbolic simplification takes a general-purpose dynamics algorithm, and applies it symbolically to a specific dynamic model. The result is a list of assignment statements detailing what the algorithm would have done if it had been executed for real. This list is then inspected and pruned of all unnecessary calculations, and the remainder is output to a text file in the form of computer source code. This code can run as much as ten times faster than the original general-purpose algorithm, but it is specific to one dynamic model. Both Autolev [3.67] and SD/FAST [3.77] use this technique. Other publications on symbolic simplification for dynamics include [3.76, 80, 81, 83–88].

3.7.8 Algorithms for Parallel Computers

In order to speed up the common dynamics computations, a number of algorithms have been developed for parallel and pipelined computers. For *inverse dynamics*, early work focused on speeding up the $O(n)$ RNEA on up to n processors [3.89, 90] while subsequent work resulted in $O(\log_2 n)$ algorithms [3.91, 92]. For the $O(n^2)$ CRBA to compute the *joint-space inertia matrix*, early work resulted in $O(\log_2 n)$ algorithms for n processors to compute the composite-rigid-body inertias and diagonal elements of the matrix [3.93, 94]. Subsequent work resulted in $O(\log_2 n)$ algorithms for $O(n^2)$ processors to compute the entire matrix [3.95, 96]. For *forward dynamics*, speedup was obtained for a multiple manipulator system on a parallel/pipelined supercomputer [3.97]. The first $O(\log_2 n)$ algorithm for n processors was developed for an unbranched serial chain [3.98]. More recent work has focused on $O(\log_2 n)$ algorithms for more complex structures [3.65, 99, 100].

3.7.9 Topologically-Varying Systems

There are many robot mechanisms whose topology varies over time because of a change of contact conditions, especially with the environment. In legged vehicles, use of a compliant ground-contact model to compute the contact forces reduced the closed-loop structure to a tree structure [3.101]. However, for cases in which the contacts are very stiff, numerical integration problems may result. In more recent work [3.40, 102] in which hard contact constraints are assumed, an efficient method was used to reduce the large number of coordinate variables from that which may be necessary in general-purpose motion analysis systems [3.43]. Also, they were able to automatically identify the variables as the structure varied and developed a method for computing the velocity boundary conditions after configuration changes [3.40, 102].

References

3.1 R. Featherstone: The calculation of robot dynamics using articulated-body inertias, Int. J. Robotics Res. **2**(1), 13–30 (1983)

3.2 J.J. Craig: *Introduction to Robotics: Mechanics and Control*, 3rd edn. (Prentice Hall, Upper Saddle River 2005)

3.3 R.E. Roberson, R. Schwertassek: *Dynamics of Multibody Systems* (Springer, Berlin, Heidelberg 1988)

3.4 J.Y.S. Luh, M.W. Walker, R.P.C. Paul: On-line computational scheme for mechanical manipulators, Trans. ASME J. Dyn. Syst. Meas. Control **102**(2), 69–76 (1980)

3.5 M.W. Walker, D.E. Orin: Efficient dynamic computer simulation of robotic mechanisms, Trans. ASME J. Dyn. Syst. Meas. Control **104**, 205–211 (1982)

3.6 D. Baraff: Linear-time dynamics using lagrange multipliers, Proc. 23rd Annu. Conf. Comp. Graph. Interact. Tech., New Orleans (1996) pp. 137–146

3.7 J. Baumgarte: Stabilization of constraints and integrals of motion in dynamical systems, Comput. Methods Appl. Mech. Eng. **1**, 1–16 (1972)

3.8 R. Featherstone: *Rigid Body Dynamics Algorithms* (Springer, New York 2008)

3.9 R.M. Murray, Z. Li, S.S. Sastry: *A Mathematical Introduction to Robotic Manipulation* (CRC, Boca Raton 1994)

3.10 J. Angeles: *Fundamentals of Robotic Mechanical Systems*, 2nd edn. (Springer, New York 2003)

3.11 R.S. Ball: *A Treatise on the Theory of Screws* (Cambridge Univ. Press, London 1900), Republished (1998)

3.12 J.M. Selig: *Geometrical Methods in Robotics* (Springer, New York 1996)

3.13 D.T. Greenwood: *Principles of Dynamics* (Prentice-Hall, Englewood Cliffs 1988)

3.14 F.C. Moon: *Applied Dynamics* (Wiley, New York 1998)

3.15 R. Featherstone: *Robot Dynamics Algorithms* (Kluwer, Boston 1987)

3.16 R. Featherstone: *Spatial v2*, http://royfeatherstone.org/spatial/v2 (2012)

3.17 S. McMillan, D.E. Orin: Efficient computation of articulated-body inertias using successive axial screws, IEEE Trans. Robotics Autom. **11**, 606–611 (1995)

3.18 L. Sciavicco, B. Siciliano: *Modeling and Control of Robot Manipulators*, 2nd edn. (Springer, London 2000)

3.19 J. Slotine, W. Li: On the adaptive control of robot manipulators, Int. J. Robotics Res. **6**(3), 49–59 (1987)

3.20 K.S. Chang, O. Khatib: Operational space dynamics: Efficient algorithms for modeling and control of branching mechanisms, Proc. IEEE Int. Conf. Robotics Autom., San Francisco (2000) pp. 850–856

3.21 O. Khatib: A unified approach to motion and force control of robot manipulators: The operational space formulation, IEEE J. Robotics Autom. **3**(1), 43–53 (1987)

3.22 Y.F. Zheng, H. Hemami: Mathematical modeling of a robot collision with its environment, J. Robotics Syst. **2**(3), 289–307 (1985)

3.23 W. Khalil, E. Dombre: *Modeling, Identification and Control of Robots* (Kogan Page Sci., London 2002)

3.24 J. Denavit, R.S. Hartenberg: A kinematic notation for lower-pair mechanisms based on matrices, J. Appl. Mech. **22**, 215–221 (1955)

3.25 H. Brandl, R. Johanni, M. Otter: A very efficient algorithm for the simulation of robots and similar multibody systems without inversion of the mass matrix, Proc. IFAC/IFIP/IMACS Int. Symp. Theory Robots, Vienna (1986)

3.26 R. Featherstone: Efficient factorization of the joint space inertia matrix for branched kinematic trees, Int. J. Robotics Res. **24**(6), 487–500 (2005)

3.27 R. Featherstone: An empirical study of the joint space inertia matrix, Int. J. Robotics Res. **23**(9), 859–871 (2004)

3.28 K. Kreutz-Delgado, A. Jain, G. Rodriguez: Recursive formulation of operational space control, Proc. IEEE Int. Conf. Robotics Autom., Sacramento (1991) pp. 1750–1753

3.29 K.W. Lilly: *Efficient Dynamic Simulation of Robotic Mechanisms* (Kluwer, Boston 1993)

3.30 K.W. Lilly, D.E. Orin: Efficient O(N) recursive computation of the operational space inertia matrix, IEEE Trans. Syst. Man Cybern. **23**(5), 1384–1391 (1993)

3.31 G. Rodriguez, A. Jain, K. Kreutz-Delgado: Spatial operator algebra for multibody system dynamics, J. Astronaut. Sci. **40**(1), 27–50 (1992)

3.32 R. Featherstone: Exploiting sparsity in operational-space dynamics, Int. J. Robotics Res. **29**(10), 1353–1368 (2010)

3.33 P. Wensing, R. Featherstone, D.E. Orin: A reduced-order recursive algorithm for the computation of the operational-space inertia matrix, Proc. IEEE Int. Conf. Robotics Autom., St. Paul (2012) pp. 4911–4917

3.34 R.E. Ellis, S.L. Ricker: Two numerical issues in simulating constrained robot dynamics, IEEE Trans. Syst. Man Cybern. **24**(1), 19–27 (1994)

3.35 J. Wittenburg: *Dynamics of Systems of Rigid Bodies* (Teubner, Stuttgart 1977)

3.36 R. Featherstone, D.E. Orin: Robot dynamics: Equations and algorithms, Proc. IEEE Int. Conf. Robotics Autom., San Francisco (2000) pp. 826–834

3.37 C.A. Balafoutis, R.V. Patel: *Dynamic Analysis of Robot Manipulators: A Cartesian Tensor Approach* (Kluwer, Boston 1991)

3.38 A. Jain: *Robot and Multibody Dynamics: Analysis and Algorithms* (Springer, New York 2011)

3.39 L.W. Tsai: *Robot Analysis and Design: The Mechanics of Serial and Parallel Manipulators* (Wiley, New York 1999)

3.40 K. Yamane: *Simulating and Generating Motions of Human Figures* (Springer, Berlin, Heidelberg 2004)

3.41 F.M.L. Amirouche: *Fundamentals of Multibody Dynamics: Theory and Applications* (Birkhäuser, Boston 2006)

3.42 M.G. Coutinho: *Dynamic Simulations of Multibody Systems* (Springer, New York 2001)

3.43 E.J. Haug: *Computer Aided Kinematics and Dynamics of Mechanical Systems* (Allyn and Bacon, Boston 1989)

3.44 R.L. Huston: *Multibody Dynamics* (Butterworths, Boston 1990)

3.45 A.A. Shabana: *Computational Dynamics*, 2nd edn. (Wiley, New York 2001)

3.46 V. Stejskal, M. Valášek: *Kinematics and Dynamics of Machinery* (Marcel Dekker, New York 1996)

3.47 L. Brand: *Vector and Tensor Analysis*, 4th edn. (Wiley/Chapman Hall, New York/London 1953)

3.48 F.C. Park, J.E. Bobrow, S.R. Ploen: A lie group formulation of robot dynamics, Int. J. Robotics Res. **14**(6), 609–618 (1995)

3.49 M.E. Kahn, B. Roth: The near minimum-time control of open-loop articulated kinematic chains, J. Dyn. Syst. Meas. Control **93**, 164–172 (1971)

3.50 J.J. Uicker: Dynamic force analysis of spatial linkages, Trans. ASME J. Appl. Mech. **34**, 418–424 (1967)
3.51 A. Jain: Unified formulation of dynamics for serial rigid multibody systems, J. Guid. Control Dyn. **14**(3), 531–542 (1991)
3.52 G. Rodriguez: Kalman filtering, smoothing, and recursive robot arm forward and inverse dynamics, IEEE J. Robotics Autom. **RA-3**(6), 624–639 (1987)
3.53 G. Rodriguez, A. Jain, K. Kreutz-Delgado: A spatial operator algebra for manipulator modelling and control, Int. J. Robotics Res. **10**(4), 371–381 (1991)
3.54 J.M. Hollerbach: A recursive lagrangian formulation of manipulator dynamics and a comparative study of dynamics formulation complexity, IEEE Trans. Syst. Man Cybern. **SMC-10**(11), 730–736 (1980)
3.55 M.W. Spong, S. Hutchinson, M. Vidyasagar: *Robot Modeling and Control* (Wiley, Hoboken 2006)
3.56 K.W. Buffinton: Kane's Method in Robotics. In: *Robotics and Automation Handbook*, ed. by T.R. Kurfess (CRC, Boca Raton 2005), 6-1–6-31
3.57 T.R. Kane, D.A. Levinson: The use of kane's dynamical equations in robotics, Int. J. Robotics Res. **2**(3), 3–21 (1983)
3.58 C.A. Balafoutis, R.V. Patel, P. Misra: Efficient modeling and computation of manipulator dynamics using orthogonal cartesian tensors, IEEE J. Robotics Autom. **4**, 665–676 (1988)
3.59 X. He, A.A. Goldenberg: An algorithm for efficient computation of dynamics of robotic manipulators, Proc. 4th Int. Conf. Adv. Robotics, Columbus (1989) pp. 175–188
3.60 W. Hu, D.W. Marhefka, D.E. Orin: Hybrid kinematic and dynamic simulation of running machines, IEEE Trans. Robotics **21**(3), 490–497 (2005)
3.61 C.A. Balafoutis, R.V. Patel: Efficient computation of manipulator inertia matrices and the direct dynamics problem, IEEE Trans. Syst. Man Cybern. **19**, 1313–1321 (1989)
3.62 K.W. Lilly, D.E. Orin: Alternate formulations for the manipulator inertia matrix, Int. J. Robotics Res. **10**, 64–74 (1991)
3.63 S. McMillan, D.E. Orin: Forward dynamics of multilegged vehicles using the composite rigid body method, Proc. IEEE Int. Conf. Robotics Autom. (1998) pp. 464–470
3.64 U.M. Ascher, D.K. Pai, B.P. Cloutier: Forward dynamics: Elimination methods, and formulation stiffness in robot simulation, Int. J. Robotics Res. **16**(6), 749–758 (1997)
3.65 R. Featherstone: A divide-and-conquer articulated-body algorithm for parallel $O(\log(n))$ calculation of rigid-body dynamics. Part 2: Trees, loops and accuracy, Int. J. Robotics Res. **18**(9), 876–892 (1999)
3.66 MSC Software Corporation: Adams, http://www.mscsoftware.com/
3.67 T. Kane, D. Levinson: *Autolev user's manual* (OnLine Dynamics Inc., Sunnyvale 2005)
3.68 Real-Time Physics Simulation: Bullet, http://bulletphysics.org/wordpress (2015)
3.69 Georgia Tech Graphics Lab and Humanoid Robotics Lab: DART, http://dartsim.github.io (2011)
3.70 S. McMillan, D.E. Orin, R.B. McGhee: DynaMechs: An object oriented software package for efficient dynamic simulation of underwater robotic vehicles. In: *Underwater Robotic Vehicles: Design and Control*, ed. by J. Yuh (TSI Press, Albuquerque 1995) pp. 73–98
3.71 Open Source Robotics Foundation: Gazebo, http://gazebosim.org (2002)
3.72 R. Smith: Open Dynamics Engine User Guide, http://opende.sourceforge.net (2006)
3.73 Microsoft Corporation: Robotics Developer Studio, http://www.microsoft.com/robotics (2010)
3.74 P.I. Corke: A robotics toolbox for MATLAB, IEEE Robotics Autom. Mag. **3**(1), 24–32 (1996)
3.75 Robotran: http://www.robotran.be (Center for Research in Mechatronics, Université catholique de Louvain 2015)
3.76 J.C. Samin, P. Fisette: *Symbolic Modeling of Multibody Systems* (Kluwer, Dordrecht 2003)
3.77 M.G. Hollars, D.E. Rosenthal, M.A. Sherman: *SD/FAST User's Manual* (Symbolic Dynamics Inc., Mountain View 1994)
3.78 M. Sherman, P. Eastman: Simbody, https://simtk.org/home/simbody (2015)
3.79 G.D. Wood, D.C. Kennedy: *Simulating Mechanical Systems in Simulink with SimMechanics* (MathWorks Inc., Natick 2003)
3.80 W. Khalil, D. Creusot: SYMORO+: A system for the symbolic modeling of robots, Robotica **15**, 153–161 (1997)
3.81 W. Khalil, A. Vijayalingam, B. Khomutenko, I. Mukhanov, P. Lemoine, G. Ecorchard: OpenSYMORO: An open-source software package for symbolic modelling of robots, Proc. IEEE/ASME Int. Conf. Adv. Intell. Mechatron. (2014) pp. 126–1211
3.82 Cyberbotics Ltd.: *Webots User Guide*, http://www.cyberbotics.com (2015)
3.83 I.C. Brown, P.J. Larcombe: A survey of customised computer algebra programs for multibody dynamic modelling. In: *Symbolic Methods in Control System Analysis and Design*, ed. by N. Munro (Inst. Electr. Eng., London 1999) pp. 53–77
3.84 J.J. Murray, C.P. Neuman: ARM: An algebraic robot dynamic modeling program, Proc. IEEE Int. Conf. Robotics Autom., Atlanta (1984) pp. 103–114
3.85 J.J. Murray, C.P. Neuman: Organizing customized robot dynamic algorithms for efficient numerical evaluation, IEEE Trans. Syst. Man Cybern. **18**(1), 115–125 (1988)
3.86 F.C. Park, J. Choi, S.R. Ploen: Symbolic formulation of closed chain dynamics in independent coordinates, Mech. Mach. Theory **34**, 731–751 (1999)
3.87 M. Vukobratovic, N. Kircanski: *Real-Time Dynamics of Manipulation Robots* (Springer, New York 1985)
3.88 J. Wittenburg, U. Wolz: Mesa Verde: A symbolic program for nonlinear articulated-rigid-body dynamics, ASME Des. Eng. Div. Conf., Cincinnati (1985) pp. 1–8, ASME Paper No. 85-DET-151
3.89 J.Y.S. Luh, C.S. Lin: Scheduling of parallel computation for a computer-controlled mechanical manipulator, IEEE Trans. Syst. Man Cybern. **12**(2), 214–234 (1982)

Avidbots Ex1016
Page 112 of 2154

3.90 D.E. Orin: Pipelined approach to inverse plant plus jacobian control of robot manipulators, Proc. IEEE Int. Conf. Robotics Autom., Atlanta (1984) pp. 169–175

3.91 R.H. Lathrop: Parallelism in manipulator dynamics, Int. J. Robotics Res. **4**(2), 80–102 (1985)

3.92 C.S.G. Lee, P.R. Chang: Efficient parallel algorithm for robot inverse dynamics computation, IEEE Trans. Syst. Man Cybern. **16**(4), 532–542 (1986)

3.93 M. Amin-Javaheri, D.E. Orin: Systolic architectures for the manipulator inertia matrix, IEEE Trans. Syst. Man Cybern. **18**(6), 939–951 (1988)

3.94 C.S.G. Lee, P.R. Chang: Efficient parallel algorithms for robot forward dynamics computation, IEEE Trans. Syst. Man Cybern. **18**(2), 238–251 (1988)

3.95 M. Amin-Javaheri, D.E. Orin: Parallel algorithms for computation of the manipulator inertia matrix, Int. J. Robotics Res. **10**(2), 162–170 (1991)

3.96 A. Fijany, A.K. Bejczy: A class of parallel algorithms for computation of the manipulator inertia matrix, IEEE Trans. Robotics Autom. **5**(5), 600–615 (1989)

3.97 S. McMillan, P. Sadayappan, D.E. Orin: Parallel dynamic simulation of multiple manipulator systems: Temporal versus spatial methods, IEEE Trans. Syst. Man Cybern. **24**(7), 982–990 (1994)

3.98 A. Fijany, I. Sharf, G.M.T. D'Eleuterio: Parallel $O(\log N)$ algorithms for computation of manipulator forward dynamics, IEEE Trans. Robotics Autom. **11**(3), 389–400 (1995)

3.99 R. Featherstone: A divide-and-conquer articulated-body algorithm for parallel $O(\log(n))$ calculation of rigid-body dynamics. Part 1: Basic algorithm, Int. J. Robotics Res. **18**(9), 867–875 (1999)

3.100 R. Featherstone, A. Fijany: A technique for analyzing constrained rigid-body systems and its application to the constraint force algorithm, IEEE Trans. Robotics Autom. **15**(6), 1140–1144 (1999)

3.101 P.S. Freeman, D.E. Orin: Efficient dynamic simulation of a quadruped using a decoupled tree-structured approach, Int. J. Robotics Res. **10**, 619–627 (1991)

3.102 Y. Nakamura, K. Yamane: Dynamics computation of structure-varying kinematic chains and its application to human figures, IEEE Trans. Robotics Autom. **16**(2), 124–134 (2000)

Multimedia Contents

4. Mechanism and Actuation

Victor Scheinman, J. Michael McCarthy, Jae-Bok Song

This chapter focuses on the principles that guide the design and construction of robotic systems. The kinematics equations and Jacobian of the robot characterize its range of motion and mechanical advantage, and guide the selection of its size and joint arrangement. The tasks a robot is to perform and the associated precision of its movement determine detailed features such as mechanical structure, transmission, and actuator selection. Here we discuss in detail both the mathematical tools and practical considerations that guide the design of mechanisms and actuation for a robot system.

The following sections (Sect. 4.1) discuss characteristics of the mechanisms and actuation that affect the performance of a robot. Sections 4.2–4.6 discuss the basic features of a robot manipulator and their relationship to the mathematical model that is used to characterize its performance. Sections 4.7 and 4.8 focus on the details of the structure and actuation of the robot and how they combine to yield various types of robots. The final Sect. 4.9 relates these design features to various performance metrics.

4.1 **Overview** 68

4.2 **System Features** 68
- 4.2.1 Work Envelope 68
- 4.2.2 Load Capacity 68
- 4.2.3 Kinematic Skeleton 69

4.3 **Kinematics and Kinetics** 69
- 4.3.1 Robot Topology 69
- 4.3.2 Kinematics Equations 71
- 4.3.3 Configuration Space 71
- 4.3.4 Speed Ratios 71
- 4.3.5 Mechanical Advantage 72

4.4 **Serial Robots** 72
- 4.4.1 Design Optimization 73
- 4.4.2 Speed Ratios 73

4.5 **Parallel Robots** 73
- 4.5.1 Workspace 74
- 4.5.2 Mechanical Advantage 74
- 4.5.3 Specialized Parallel Robots 75

4.6 **Mechanical Structure** 75
- 4.6.1 Links 75
- 4.6.2 Joints 76

4.7 **Joint Mechanisms** 76
- 4.7.1 Joint Axis Structures 76

4.8 **Actuators** 78
- 4.8.1 Electromagnetic Actuators 78
- 4.8.2 Hydraulic Actuators 80
- 4.8.3 Pneumatic Actuators 81
- 4.8.4 Other Actuators 82
- 4.8.5 Transmissions 82

4.9 **Robot Performance** 85
- 4.9.1 Robot Speed 85
- 4.9.2 Robot Acceleration 85
- 4.9.3 Repeatability 85
- 4.9.4 Resolution 86
- 4.9.5 Accuracy 86
- 4.9.6 Component Life and Duty Cycle 86
- 4.9.7 Collisions 86

4.10 **Conclusions and Further Reading** 87

Video-References 87

References 87

4.1 Overview

The physical structure such as the beams, links, castings, shafts, slides, and bearings of a robot that create its movable skeleton is termed the mechanical structure or mechanism of the robot. The electric, hydraulic and pneumatic motors and other elements that cause the links of the mechanism to move are called actuators. In this chapter we consider the variety of designs for the mechanisms and actuators that result in a machine system that transforms computer commands into versatile physical movement.

Early robots were designed with general motion capability under the assumption that they would find the largest market if they could perform the widest variety of tasks; this emphasis on flexibility proved to be expensive in both cost and performance. Robots are now often designed for specific applications and to perform limited sets of tasks.

Robot design focuses on the number of joints, physical size, payload capacity, and the movement requirements of the end-effector. The configuration of the movable skeleton and the overall size of the robot are determined by task requirements for reach, workspace, and reorientation ability. These features affect the precision of end-effector path control needed for applications such as for arc-welding and for the smooth movement of paint spraying. They also define the absolute positioning capability necessary for small part assembly, the repeatability needed for material and package handling, and the fine resolution that allows precise, real-time sensor-based motions.

A critical concern in robotic system design is the range of tasks the robot is expected to perform. The robot should be designed to have the flexibility it needs to perform the range of tasks for which it is intended. This determines the topology of the robot mechanism and the actuator system. The choices of geometry, material, sensors, and cable routing follow from these basic decisions.

4.2 System Features

The primary features that characterize a robot are its work envelope and load capacity.

4.2.1 Work Envelope

The space in which a robot can operate is its work envelope, which encloses its workspace. While the workspace of the robot defines positions and orientations that it can achieve to accomplish a task, the work envelope also includes the volume of space the robot itself occupies as it moves. This envelope is defined by the types of joints, their range of movement and the lengths of the links that connect them. The physical size of this envelope and the loads on the robot within this envelope are of primary consideration in the design of the mechanical structure of a robot.

Robot work envelope layouts must include considerations of regions of limited accessibility where the mechanical structure may experience movement limitations. These constraints arise from limited joint travel range, link lengths, the angles between axes, or a combination of these. Revolute joint manipulators generally work better in the middle of their work envelopes than at extremes (Fig. 4.1). Manipulator link lengths and joint travel should be chosen to leave margins for variable sensor-guided controlled path motions and for tool or end-effector changes, as offsets and length differences will often alter the work envelope.

4.2.2 Load Capacity

Load capacity, a primary robot specification, is closely coupled with acceleration and speed. For assembly robots, mechanism acceleration and stiffness (structure and drive stiffness) are often more important design parameters than peak velocity or maximum load capacity, as minimizing pick-and-place motion cycle time, while maintaining placement precision, is generally a top priority in small part assembly. In the case of arc-welding, where slow controlled-path motion is required, velocity

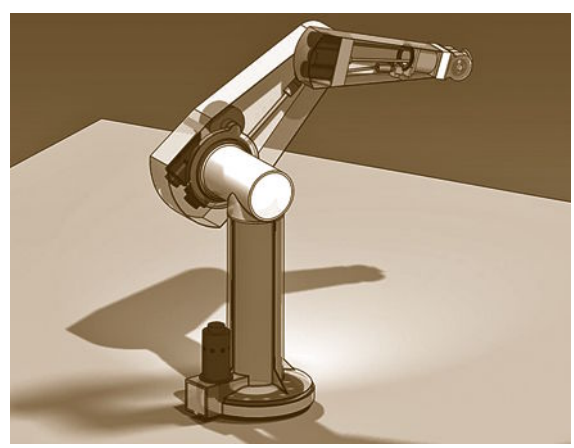


Fig. 4.1 The PUMA 560 robot

jitter and weld path-following accuracy are important. Load capacity should be seen as a variable. It is wise to design and specify a manipulator in terms of useful payload as a function of performance rather than just in terms of maximum capacity.

Load capacity specifications must take into account gravity and inertial loading seen at the end-effector. These factors strongly affect wrist, end-effector design and drive selection. In general, load capacity is more a function of manipulator acceleration and wrist torque than any other factor. The load rating also affects manipulator static structural deflection, steady-state motor torque, system natural frequency, damping, and the choice of control system parameters for best performance and stability.

4.2.3 Kinematic Skeleton

Manipulator shape and size is determined by requirements on its workspace shape and layout, the precision of its movement, its acceleration and speed, and its construction. Cartesian manipulators have the simplest transform and control equation solutions. Their prismatic (straight-line motion) and orthogonal (perpendicular) axes make motion planning and computation relatively straightforward. Because their major motion axes do not couple dynamically (to a first order), their control equations are also simplified. Manipulators with all revolute (rotary) joints are generally harder to control, but they feature a more compact and efficient structure for a given working volume. It is generally easier to design and build a good revolute joint than a long motion prismatic joint. The workspaces of revolute joint manipulators can easily overlap for coordinated multi-robot installations, in contrast to the more exclusive useful workspace of gantry style robots.

Final selection of the robot configuration should capitalize on specific kinematic, structural or performance requirements. For example, a requirement for a very precise vertical straight-line motion may dictate the choice of a simple prismatic vertical axis rather than two or three revolute joints requiring coordinated control.

Six degrees of freedom (DOFs) are the minimum required to place the end-effector or tool of a robotic manipulator at any arbitrary location (position and orientation) within its accessible workspace. Most simple or preplanned tasks can be performed with fewer than six DOFs by careful task setup to eliminate certain axis motions, or because the tool or task does not require full specification of location. An example of this is vertical assembly using a powered screwdriver, where all operations can be achieved with three degrees of freedom.

Some applications require the use of manipulators with more than six DOFs, in particular when mobility or obstacle avoidance are necessary. For example, a pipe-crawling maintenance robot requires control of the robot shape as well as precise positioning of its end-effector. Generally, adding degrees of freedom increases cycle time and reduces load capacity and accuracy for a given manipulator configuration and drive system.

4.3 Kinematics and Kinetics

The dynamics of a robot can be separated into the properties of the movement that depend upon the geometry of its mechanical structure, termed *kinematics*, and those that depend on forces that act on the system, known as *kinetics*. It is a law of dynamics that the difference between the change in energy of the moving robot and the work performed by the forces acting on it does not change over small variations of its trajectory. This is called the *principle of virtual work* and states that variations in work and energy must cancel for all virtual displacements [4.1, 2].

Because machines such as robots are designed to minimize energy losses, often due to joint friction and material strain losses, we can assume that the variation in energy is small. This means that the work input of the actuators is nearly equal to the work of the output forces.

If we consider this relationship over a small duration of time, we have that the time rate of input work, or input power is nearly equal to the associated output power. Because power is force times velocity, we obtain the fundamental relationship that the ratio of input to output forces is the reciprocal of the ratio of input to output speeds. Another way of saying this is that, in the ideal machine, *the mechanical advantage of a machine is the inverse of its speed ratio.*

4.3.1 Robot Topology

The kinematic skeleton of a robot is modeled as a series of links connected by either hinged or sliding joints forming a serial chain. This skeleton has two basic forms, that of a single serial chain called a *serial robot* (Fig. 4.1) and as a set of serial chains supporting a single end-effector, called a *parallel robot*, such as the platform shown in Fig. 4.2. Robots can be configured to work in parallel such as the individual legs of walking machines (Figs. 4.3 and 4.4) [4.3], as

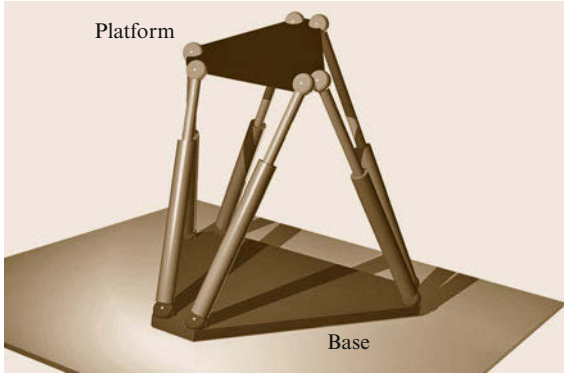


Fig. 4.2 A parallel robot can have as many as six serial chains that connect a platform to the base frame (VIDEO 640)



Fig. 4.3 A photograph of the adaptive suspension vehicle (ASV)walking machine

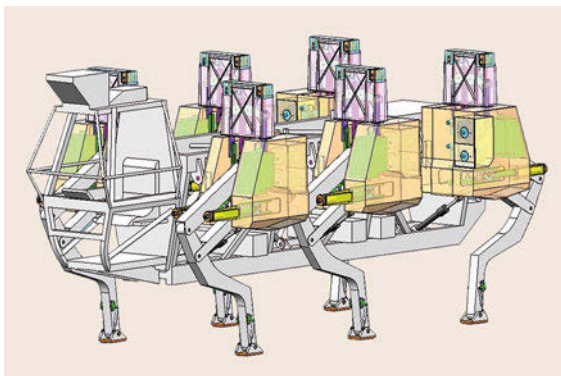


Fig. 4.4 The adaptive suspension vehicle walking machine

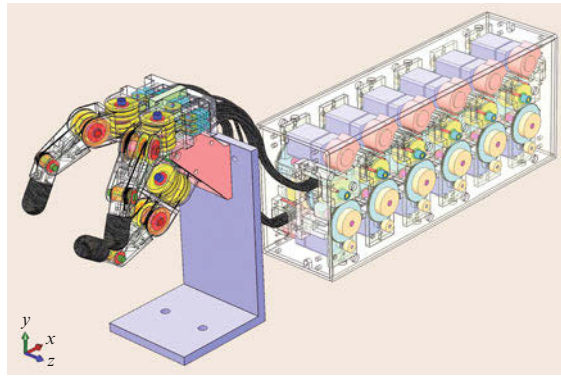


Fig. 4.5 The Salisbury three-fingered robot hand with its cable drive system

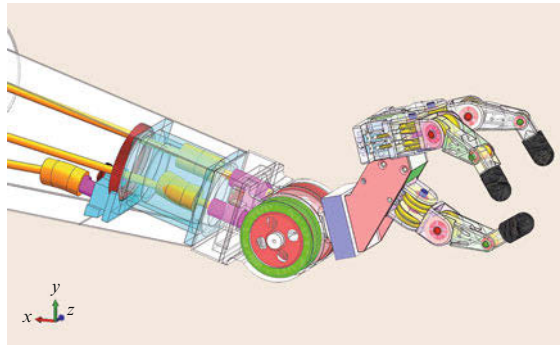


Fig. 4.6 The Salisbury hand as the end-effector of a PUMA robot (the drive system is not shown)

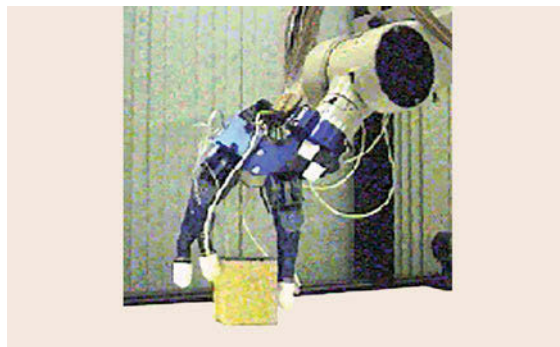


Fig. 4.7 A photograph of the Salisbury three-fingered hand grasping a block

well as the fingers of mechanical hands (Figs. 4.5–4.7 and VIDEO 642) [4.4].

The robot end-effector is the preferred tool for interaction with the environment, and the ability to position and orient this end-effector is defined by the skeleton of the robot. In a general serial robot, a chain of six joints provides full control over the end-effector. In a general parallel robot, there are more than six joints and the six actuators may be applied to these joints in a variety of ways to control the movement of the end-effector.

Avidbots Ex1016
Page 117 of 2154

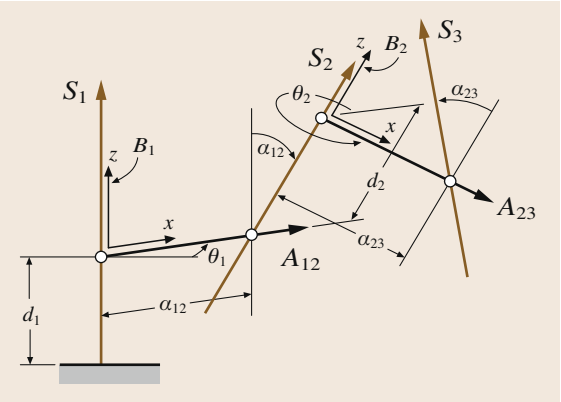


Fig. 4.8 The kinematic skeleton defined by the join axes and their common normals

4.3.2 Kinematics Equations

A robot is designed so that specifying the values of local joint parameters, such as the angle of rotary joints and the travel of sliding joints, specifies the position of every component of a machine using its kinematics equations. To do this the robot is described by a sequence of lines representing the axes $\hat{z}_j$ of equivalent revolute or prismatic joints and the common normal lines $\hat{x}_j$ which form the kinematic skeleton of the chain (Fig. 4.8). This construction allows the specification of the location of each link of the robot relative to the base by the matrix equation

$$\mathbf{T} = \mathbf{Z}(\theta_1, d_1)\mathbf{X}(\alpha_1, a_1)\mathbf{Z}(\theta_2, d_2)\ldots \times \mathbf{X}(\alpha_{m-1}, a_{m-1})\mathbf{Z}(\theta_m, d_m)\,, \tag{4.1}$$

known as the *kinematics equations* of the chain [4.5, 6] (Chap. 2; (2.46)). The set of all positions $\mathbf{T}$ that the end-effector can reach for all values of the joint parameters is called the *workspace* of the robot.

The matrices $\mathbf{Z}(\theta_j, d_j)$ and $\mathbf{X}(\alpha_j, a_j)$ are 4×4 matrices that define screw displacements around and along the joint axes $\hat{z}_j$ and $\hat{x}_j$, respectively [4.7]. The parameters α_j and a_j define the dimensions of the links in the chain. The parameter θ_j is the joint variable for revolute joints and d_j is the variable for prismatic joints. The trajectory ${}^F\boldsymbol{p}(t)$ of a point ${}^M\boldsymbol{p}$ in the end-effector is obtained from the joint trajectory,

$$\boldsymbol{q}(t) = (q_1(t), \ldots, q_m(t))^{\mathrm{T}}\,,$$

where q_i is either θ_i or d_i depending on the joint, given by

$${}^F\boldsymbol{p}(t) = \mathbf{T}(\boldsymbol{q}(t)){}^M\boldsymbol{p}\,. \tag{4.2}$$

If the end-effector is connected to the base frame by more than one serial chain (Fig. 4.2) then we have a set of kinematics equations for each chain,

$$\mathbf{T} = \mathbf{B}_j\mathbf{T}(\boldsymbol{q}_j)\mathbf{E}_j, \quad j = 1, \ldots, n\,, \tag{4.3}$$

where $\mathbf{B}_j$ locates the base of the j-th chain and $\mathbf{E}_j$ defines the position of its attachment to the part, or end-effector. The set of positions $\mathbf{T}$ that simultaneously satisfy all of these equations is the workspace of the part. This imposes constraints on the joint variables that must be determined to define its workspace completely [4.8, 9].

4.3.3 Configuration Space

The kinematics equations of the robot relate the range of values available for the joint parameters, called the configuration space of the robot, to the workspace of the end-effector. This configuration space is a fundamental tool in robot path planning for obstacle avoidance [4.10]. Though any link in the chain forming a robot may hit an obstacle, it is the end-effector that is intended to approach and move around obstacles such as the table supporting the robot and the fixtures for parts it is to pick up. Obstacles define forbidden positions and orientations in the workspace which map back to forbidden joint angles in the configuration space of the robot. Robot path planners seek trajectories to a goal position through the free space around these joint space obstacles [4.11].

4.3.4 Speed Ratios

The speed ratios of a robot relate the velocity ${}^F\dot{\boldsymbol{p}}$ of a point ${}^F\boldsymbol{p}$ in the end-effector to the joint rates

$$\dot{\boldsymbol{q}} = (\dot{q}_1, \ldots, \dot{q}_m)^{\mathrm{T}}\,,$$

that is

$${}^F\dot{\boldsymbol{p}} = \boldsymbol{v} + \boldsymbol{\omega} \times \left({}^F\boldsymbol{p} - \boldsymbol{d}\right), \tag{4.4}$$

where $\boldsymbol{d}$ and $\boldsymbol{v}$ are the position and velocity of a reference point, respectively, and $\boldsymbol{\omega}$ is the angular velocity of the end-effector.

The vectors $\boldsymbol{v}$ and $\boldsymbol{\omega}$ depend on the joint rates $\dot{q}_j$ through the formula

$$\begin{pmatrix}\boldsymbol{v}\\ \boldsymbol{\omega}\end{pmatrix} = \begin{pmatrix}\frac{\partial \boldsymbol{v}}{\partial \dot{q}_1} & \frac{\partial \boldsymbol{v}}{\partial \dot{q}_2} & \cdots & \frac{\partial \boldsymbol{v}}{\partial \dot{q}_m}\\ \frac{\partial \boldsymbol{\omega}}{\partial \dot{q}_1} & \frac{\partial \boldsymbol{\omega}}{\partial \dot{q}_2} & \cdots & \frac{\partial \boldsymbol{\omega}}{\partial \dot{q}_m}\end{pmatrix}\begin{pmatrix}\dot{q}_1\\ \vdots\\ \dot{q}_m\end{pmatrix}, \tag{4.5}$$

Avidbots Ex1016
Page 118 of 2154

or

$$\mathbf{v} = \mathbf{J}\dot{\boldsymbol{q}} \,. \tag{4.6}$$

The coefficient matrix $\mathbf{J}$ in this equation is called the *Jacobian* and is a matrix of speed ratios relating the velocity of the tool to the input joint rotation rates [4.6, 9].

4.3.5 Mechanical Advantage

If the end-effector of the robot exerts a force $\boldsymbol{f}$ at the point ${}^F\boldsymbol{p}$, then the power output is

$$P_{\text{out}} = \boldsymbol{f} \cdot {}^F\dot{\boldsymbol{p}} = \sum_{j=1}^{m} \boldsymbol{f} \cdot \left[\frac{\partial \boldsymbol{v}}{\partial \dot{q}_j} + \frac{\partial \boldsymbol{\omega}}{\partial \dot{q}_j} \times \left({}^F\boldsymbol{p} - \boldsymbol{d}\right) \right] \dot{q}_j \,. \tag{4.7}$$

Each term in this sum is the portion of the output power that can be associated with an actuator at joint S_j, if one exists.

The power input at joint S_j is the product $\tau_j \dot{q}_j$ of the torque τ_j and joint angular velocity $\dot{q}_j$. Using the principle of virtual work for each joint we can compute

$$\tau_j = \boldsymbol{f} \cdot \frac{\partial \boldsymbol{v}}{\partial \dot{q}_j} + \left({}^F\boldsymbol{p} - \boldsymbol{d}\right) \times \boldsymbol{f} \cdot \frac{\partial \boldsymbol{\omega}}{\partial \dot{q}_j}, \quad j = 1, \ldots, m \,. \tag{4.8}$$

We have arranged this equation to introduce the force-torque vector

$$\boldsymbol{f} = (\boldsymbol{f}, ({}^F\boldsymbol{p} - \boldsymbol{d}) \times \boldsymbol{f})^{\mathrm{T}}$$

at the reference point $\boldsymbol{d}$.

The equations (4.8) can be assembled into the matrix equation

$$\boldsymbol{\tau} = \mathbf{J}^{\mathrm{T}} \boldsymbol{f} \,, \tag{4.9}$$

where $\mathbf{J}$ is the *Jacobian* defined above in (4.5). For a chain with six joints this equation can be solved for the output force-torque vector $\boldsymbol{f}$,

$$\boldsymbol{f} = \left(\mathbf{J}^{\mathrm{T}}\right)^{-1} \boldsymbol{\tau} \,. \tag{4.10}$$

Thus, the matrix that defines the mechanical advantage for this system is the inverse of the matrix of speed ratios.

4.4 Serial Robots

A serial chain robot is a sequence of links and joints that begins at a base and ends with an end-effector (Fig. 4.9). The links and joints of a robot are often configured to provide separate translation and orientation structures. Usually, the first three joints are used to position a reference point in space and the last three form the *wrist* which orients the end-effector around this point [4.12, 13]. This reference point is called the *wrist center*. The volume of space in which the wrist center can be placed is called the *reachable workspace* of the robot. The rotations available at each of these points is called the *dexterous workspace*.

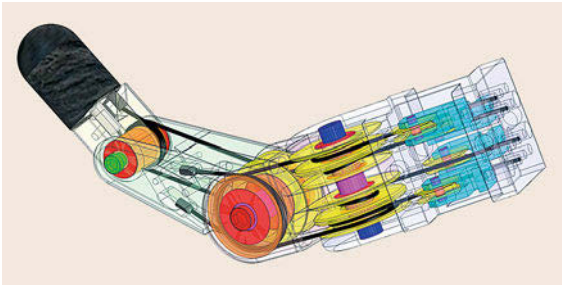


Fig. 4.9 A single finger of the Salisbury hand is a serial chain robot

The design of a robot is often based on the symmetry of its reachable workspace. From this point of view there are three basic shapes: rectangular, cylindrical, and spherical [4.6].A rectangular workspace is provided by three mutually perpendicular prismatic (P) joints which form a PPPS chain called a *Cartesian* robot – S denotes a spherical wrist which allows all rotations about its center point. A rotary base joint combined with two prismatic joints forms a CPS chain with a cylindrical workspace – C denotes a rotary (R) and sliding (P) joint with the same axis. The P-joint can be replaced by a revolute (R) joint that acts as an elbow in order to provide the same radial movement. Finally, two rotary joints at right angles form a T-joint at the base of the robot that supports rotations about a vertical and horizontal axes. Radial movement is provided either by a P-joint, or by an R-joint configured as an elbow. The result is a TPS or TRS chain with a spherical workspace.

It is rare that the workspace is completely symmetrical because joint axes are often offset to avoid link collisions and there are limits to joint travel which combine to distort the shape of the workspace.

4.4.1 Design Optimization

Another approach to robot design uses a direct specification of the workspace as a set of positions for the end-effector of a robotic system [4.14–17] which we call the *task space*. A general serial robot arm has two design parameters, link offset and twist, for each of five links combined with four parameters each that locate the base of the robot and the workpiece in its end-effector, making a total of 18 design variables. The link parameters are often specified so the chain has a spherical wrist and specific workspace shape. The design goal is usually to determine the workspace volume and locate the base and workpiece frames so that the workspace encloses the specified task space.

The task space is defined by a set of 4×4 transformations $\mathbf{D}_i, i=1,\ldots,k$. The problem is solved iteratively by selecting a design and using the associated kinematics equations $\mathbf{T}(\boldsymbol{q})$ to evaluate relative displacements in the objective function

$$f(\boldsymbol{r}) = \sum_{i=1}^{k} \left\| \mathbf{D}_i \mathbf{T}^{-1}(\boldsymbol{q}_i) \right\| \ . \tag{4.11}$$

Optimization techniques yield the design parameter vector $\boldsymbol{r}$ that minimizes this objective function.

This optimization depends on the definition of the distance measure between the positions reached by the end-effector and the desired workspace. *Park* [4.18], *Martinez* and *Duffy* [4.19], *Zefran* et al. [4.20], *Lin* and *Burdick* [4.21], and others have shown that there is no distance metric that is coordinate frame invariant. This means that, unless this objective function can be forced to zero so the workspace completely contains the task space, the resulting design will not be *geometric* in the sense that the design is not independent of the choice of coordinates.

4.4.2 Speed Ratios

A six-axis robot has a 6×6 Jacobian $\mathbf{J}$ obtained from (4.5) that is an array of speed ratios relating the components of the velocity $\boldsymbol{v}$ of the wrist center and the angular velocity $\boldsymbol{\omega}$ of the end-effector to each of the joint velocities. Equation (4.9) shows that this Jacobian defines the force-torque vector $\boldsymbol{f}$ exerted at the wrist center in terms of the torque applied by each of the actuators. The link parameters of the robot can be selected to provide a Jacobian $\mathbf{J}$ with specific properties.

The sum of the squares of the actuator torques of a robot is often used as a measure of *effort* [4.22, 23]. From (4.9) we have

$$\boldsymbol{\tau}^{\mathrm{T}}\boldsymbol{\tau} = \boldsymbol{f}^{\mathrm{T}}\mathbf{J}\mathbf{J}^{\mathrm{T}}\boldsymbol{f} \ . \tag{4.12}$$

The matrix $\mathbf{J}\mathbf{J}^{\mathrm{T}}$ is square and positive definite. Therefore, it can be viewed as defining a hyperellipsoid in six-dimensional space [4.24]. The lengths of the semi-diameters of this ellipsoid are the inverse of the absolute value of the eigenvalues of the Jacobian $\mathbf{J}$. These eigenvalues may be viewed as *modal* speed ratios that define the amplification associated with each joint velocity. Their reciprocals are the associated *modal* mechanical advantages, so the shape of this ellipsoid illustrates the force amplification properties of the robot.

The ratio of the largest of these eigenvalues to the smallest, called the condition number, gives a measure of the anisotropy or *out-of-roundness* of the ellipsoid. A sphere has a condition number of one and is termed *isotropic*. When the end-effector of a robot is in a position with an isotropic Jacobian there is no amplification of the speed ratios or mechanical advantage. This is considered to provide high-fidelity coupling between the input and output because errors are not amplified [4.25, 26]. Thus, the condition number is used as a criterion in a robot design [4.27].

In this case, it is assumed that the basic design of the robot provides a workspace that includes the task space. Parameter optimization finds the internal link parameters that yield the desired properties for the Jacobian. As in minimizing the distance to a desired workspace, optimization based on the Jacobian depends on a careful formulation to avoid coordinate dependency.

4.5 Parallel Robots

A robotic system in which two or more serial chain robots support an end-effector is called a *parallel robot*. For example, the adaptive suspension vehicle (ASV) leg (Fig. 4.10), is a pantograph mechanism driven by parallel actuation. Each supporting chain of a parallel robot may have as manyas six degrees of freedom, however, in general only a total of six joints in the entire system are actuated. A good example is the Stewart platform formed from six TPS robots in which usually only the prismatic joint (P-joint) in each chain is actuated (Fig. 4.2) [4.9, 28, 29].

The kinematics equations of the TPS legs are

$$\mathbf{T} = \mathbf{B}_j\mathbf{T}(\boldsymbol{\theta}_j)\mathbf{E}_j, \quad j=1,\ldots,6 \ , \tag{4.13}$$

Avidbots Ex1016
Page 120 of 2154

where $\mathbf{B}_j$ locates the base of the leg and $\mathbf{E}_j$ defines the position of its attachment to the end-effector. The set of positions $\mathbf{T}$ that simultaneously satisfy all of these equations is the workspace of the parallel robot.

Often the workspace of an individual chain of a parallel robot can be defined by geometric constraints, for example, a position $\mathbf{T}$ is in the workspace of the j-th supporting TPS chain if it satisfies the constraint equation

$$(\mathbf{T}\boldsymbol{x}_j - \boldsymbol{p}_j) \cdot (\mathbf{T}\boldsymbol{x}_j - \boldsymbol{p}_j) = \rho_j^2 \,. \tag{4.14}$$

This equation defines the distance between the base joint $\boldsymbol{p}_j$ and the point of attachment ${}^F\boldsymbol{x}_j = \mathbf{T}\boldsymbol{x}_j$ to the platform as the length ρ_j is controlled by the actuated prismatic joint. In this case the workspace is the set of positions $\mathbf{T}$ that satisfy all six equations, one for each leg.

4.5.1 Workspace

The workspace of a parallel robot is the intersection of the workspaces of the individual supporting chains. However, it is not the intersection of the reachable and dexterous workspaces separately. These workspaces are intimately combined in parallel robots. The dexterous workspace is usually largest near the center of the reachable workspace and shrinks as the reference point moves toward the edge. A focus on the symmetry of movement allowed by supporting leg designs has been an important design tool resulting in many novel parallel designs [4.30, 31]. Simulation of the system is used to evaluate its workspace in terms of design parameters.

Another approach is to specify directly the positions and orientations that are to lie in the workspace and solve the algebraic equations that define the leg constraints to determine the design parameters [4.32, 33]. This is known as kinematic synthesis and yields parallel robots that are asymmetric but have specified reachable and dexterous workspaces, see *McCarthy* and *Soh* [4.34].

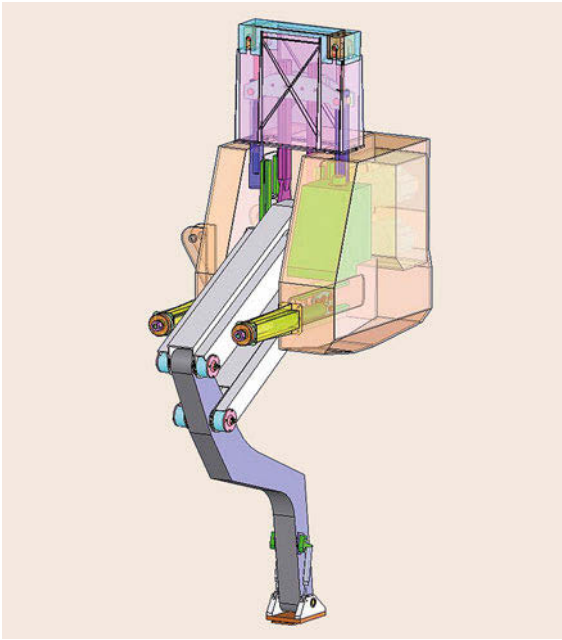


Fig. 4.10 One leg of the ASV walking machine is a parallel robot

4.5.2 Mechanical Advantage

The force amplification properties of a parallel robot are obtained by considering the Jacobians of the individual supporting chains. Let the linear and angular velocity of the platform be defined by the six-vector $\mathbf{v} = (\boldsymbol{v}, \boldsymbol{\omega})^{\mathrm{T}}$, then from the kinematics equations of each of the support legs we have

$$\mathbf{v} = \mathbf{J}_1\dot{\boldsymbol{\rho}}_1 = \mathbf{J}_2\dot{\boldsymbol{\rho}}_2 = \cdots = \mathbf{J}_6\dot{\boldsymbol{\rho}}_6 \,. \tag{4.15}$$

Here we assume that the platform is supported by six chains, but it can be fewer, such as when the fingers of a mechanical hand grasp an object [4.4].

The force on the platform applied by each chain is obtained from the principle of virtual work as

$$\boldsymbol{f}_j = \left(\mathbf{J}_j^{\mathrm{T}}\right)^{-1}\boldsymbol{\tau}_j, \quad j = 1, \ldots, 6 \,. \tag{4.16}$$

There are only six actuated joints in the system so we assemble the associated joint torques into the vector $\boldsymbol{\tau} = (\tau_1, \ldots, \tau_6)^{\mathrm{T}}$. If $\boldsymbol{f}_i$ is the force-torque vector obtained from (4.16) for $\tau_i = 1$ and the remaining torques to zero, then the resultant force-torque $\mathbf{w}$ applied to the platform is

$$\mathbf{w} = (\boldsymbol{f}_1, \boldsymbol{f}_2, \cdots, \boldsymbol{f}_6)\boldsymbol{\tau} \,, \tag{4.17}$$

or

$$\mathbf{w} = \boldsymbol{\Gamma}\boldsymbol{\tau} \,. \tag{4.18}$$

The elements of the coefficient matrix $\boldsymbol{\Gamma}$ define the mechanical advantage for each of the actuated joints. In the case of a Stewart platform the columns of this matrix are the Plücker coordinates of the lines along each leg [4.29].

The principle of virtual work yields the velocity of the platform in terms of the joints rates $\dot{\boldsymbol{\rho}}$ as

$$\boldsymbol{\Gamma}^{\mathrm{T}}\mathbf{v} = \dot{\boldsymbol{\rho}} \,. \tag{4.19}$$

Thus, the inverse of $\mathbf{\Gamma}$ defines the speed ratios between the actuated joints and the end-effector. The same equation can be obtained by computing the derivative of the geometric constraint equations (4.14), and $\mathbf{\Gamma}$ is the *Jacobian* of the parallel robot system [4.35].

The Jacobian $\mathbf{\Gamma}$ is used in parameter optimization algorithms to design parallel robots [4.36] with isotropic mechanical advantage. The square root of the determinant $\left|\mathbf{\Gamma}\mathbf{\Gamma}^{\mathrm{T}}\right|$ measures the six-dimensional volume spanned by the column vectors $\boldsymbol{f}_j$. The distribution of the percentage of this volume compared to its maximum within the workspace is also used as a measure of the overall performance [4.37, 38]. A similar performance measure normalizes this Jacobian by the maximum joint torques available and the maximum component of force and torque desired, and then seeks an isotropic design [4.39].

4.5.3 Specialized Parallel Robots

Another approach to the design of parallel robots has been to separate their functionality into orientation and translational platforms. *Tsai* and *Joshi* [4.40] and *Jin* and *Yang* [4.41] survey designs for a class of parallel chains that generate pure translation. *Kong* and *Gosselin* [4.42] and *Hess-Coelho* [4.43] do the same for parallel chains that provide rotational movement in space.

4.6 Mechanical Structure

For the purposes of dynamic modeling, the links of a robot are generally considered to be rigid. However, a robot is not a rigid structure. Like all structures it deflects under applied loads, such as its own weight and the weight of the payload, termed gravity loading; see Figs. 4.11 and 4.12. The issue is a matter of degree. The more force that is needed to cause a deflection in the links, the more the robot moves like a connected set of rigid bodies. Rigid robots have links designed to be stiff so the deflections under load are less than the positioning accuracy required for their range of tasks. This allows the dynamic model and control algorithms to ignore link deflection. Most commercially available robot arms are of this type (*Rivin* [4.44]).

It is possible to improve the positioning accuracy of a rigid robot by augmenting a control algorithm that includes a model of link deflection resulting from gravity loading. It is also possible to use strain sensors to measure loads and deflections. These *semirigid* robots assume small structural deflections that are linearly related to known applied loads.

Flexible robots require that the dynamic model include the deflection of its various links under gravity loading as well as under the forces associated with link acceleration, called inertia loading. The robot control algorithms must control the vibration of the system as well its gross motion. Management of vibration may be required even in rigid robots to reduce cycle time when subjected to high dynamic loading at high speeds or when manipulating large payloads.

4.6.1 Links

For industrial robots a critical concern is the link stiffness in bending and in torsion. To provide this stiffness, robot links are designed either as beams or shell (monocoque) structures. Monocoque structures have lower weight or higher strength-to-weight ratios, but are more costly and generally more difficult to manufacture. Cast, extruded, or machined beam-based links are often more cost effective; *Juvinall* and *Marshek* [4.45], and *Shigley* and *Mischke* [4.46, 47]

Another important consideration is whether the link structure includes bolted, welded, or adhesive bonded assemblies of cast, machined, and fabricated elements. Screw and bolted connections may seem straightforward, inexpensive, and easily maintained, but the inevitable deflection of a link even in the manufacturing process introduces creep in these multiple element assemblies that changes the dimensions and performance of the robot. Welded and cast structures are much less susceptible to creep and the associated hysteresis deformation, though in many cases they require secondary manufacturing operations such as thermal stress relieving and finish machining.

The minimum practical wall or web thickness for castings may be thicker than necessary for stiffness. Thin walls can be achieved with structural skin (mono-



Fig. 4.11 The hydraulic Skywash aircraft cleaning robot



Fig. 4.12 DeLaval Cow Milking System features hydraulic robot with machine vision guided positioning (VIDEO 643)

coque) structures but this is offset by the potential for denting, permanent deformation, and damage in the event of slight collisions. Therefore, both the performance and application requirements must be considered when selecting the construction and fabrication details of the robot.

Performance- and application-specific materials and geometry are used to reduce the weight of the links and therefore the associated gravity and inertial loading. For structures that move in a straight line, aluminum or magnesium alloy extrusions of constant cross section are convenient. Carbon and glass-fiber composites provide lower mass for robots that require high acceleration (painting robots). Thermoplastic materials provide low-cost link structures though at reduced load capacity. Stainless steel is often used in robots for medical and food service applications. The longer links on serial chain robots often are designed to taper in cross section or wall thickness to reduce the associated inertial loading.

4.6.2 Joints

Joints for most robots allow either rotary or linear movement, termed revolute and prismatic joints. Other joints that are available are the ball-in-socket, or spherical joint, and the Hooke-type universal joint.

Integration of the mechanical structure of the robot with its joint mechanism, which includes the actuator and joint motion sensor, is a source of structural flexibility. Precision is reduced when deformation in the joint at its bearings can reduce gear and shaft preloads, allowing undesirable backlash and free play. Structural flexibility can also introduce changes in gear center spacing, introducing forces and torques and associated deflection, binding, jamming, and wear.

4.7 Joint Mechanisms

A robot joint mechanism consists of at least four major components: the joint axis structure, an actuator, transmission, and state sensor (usually for position and velocity feedback, but force sensors are also common).

For low-performance manipulators that accelerate the payload at less than a peak of 0.5 g, system inertia is not as important as gravity forces and torques. This means the actuators can be placed near the joints, and their suspended weight compensated by using counterbalancing masses, springs, or gas pressure.

In high-performance robots where peak payload accelerations reach 3–10 g or more, minimizing system inertia is important. The actuators are placed near the first joint axis of a serial link manipulator to minimize its inertial contribution, and drive linkages, belts, cables or spaced gear stages are used to drive the joints.

While a longer transmission distance can reduce mass and gravity moments and inertia, it introduces flexibility and thus reduces the system stiffness. The design of the actuator placement and transmission for each joint is a trade-off between weight, inertia, stiffness, and complexity. This choice dictates the major physical characteristics of a manipulator design. To illustrate this point, consider the Adept 1 assembly robot with four degrees of freedom, each a different structure. The first axis has a direct motor drive. The second axis is driven by a steel band drive, and the third by a synchronous belt drive. Finally, the fourth axis uses a linear ball–screw drive. For a variety of useful joint mechanisms *Sclater* and *Chironis* [4.48].

4.7.1 Joint Axis Structures

Revolute Joints

Revolute or rotary motion joints are designed to perform pure rotation while minimizing other displacements and motions. The most important measure of the

quality of a revolute joint is its stiffness or resistance to all undesired motion. Key factors to be considered in design for stiffness are shaft diameter, mounting configuration and preloading of the bearings and proper clearances and tolerances. Shaft diameter and bearing size are not always based on load-carrying capacity; rather, they often will be selected to be compatible with a rigid mounting configuration and also have a bore large enough to pass cables, hoses, and even drive elements for other joints. Because joint shafts will frequently be torque-transmitting members, they and their supporting structure must be designed both for bending and torsional stiffness. The first axis of the PUMA robot is an example of such a joint with its large-diameter tubular configuration.

An important factor in maintaining stiffness in a revolute joint is the choice of bearing-mounting configuration. The mounting arrangement and mount must be designed to accommodate manufacturing tolerances, thermal expansion and bearing preload. Axial preloading of ball or tapered roller bearings improves system accuracy and stiffness by minimizing both axial and radial play. Preloads can be achieved through selective assembly or elastic (spring) elements, shim spacers, threaded collars, four-point contact bearings, duplex bearing arrangements, and tight manufacturing tolerances.

Prismatic Joints

There are two basic types of prismatic or linear motion joints: single-stage and telescoping joints. Often a prismatic joint and its associated link and actuator are combined as a linear actuator. Single-stage joints are made up of a moving surface that slides linearly along a fixed surface. Telescoping joints are essentially sets of nested or stacked single-stage joints. Single-stage joints feature simplicity and high stiffness, whereas the primary advantage of telescoping joints is their retracted-state compactness and large extension ratio. Telescoping joints have a lower effective joint inertia for some configurations and motions because part of the joint may remain stationary or move with reduced acceleration.

The primary functions of bearings in prismatic joints are to facilitate motion in a single direction and to prevent motion in all other directions. Preventing these unwanted motions poses the more challenging design problem. Deformations in the structure can distort the bearing. In severe cases, ball or roller deflection under load may cause binding, which precludes motion. For high-precision prismatic joints, ways must be straight and precise along their entire length which may be several meters. The required precision grinding on multiple surfaces can be costly. Bulky covers may be required to shield and seal a prismatic bearing and way assembly.

The primary criterion for evaluating higher number (in or near the wrist or end-effector) linear motion joints or axes is the stiffness-to-weight ratio. Achieving a good stiffness-to-weight ratio requires the use of a hollow or thin-walled structure rather than solid members for the moving elements.

Bearing spacing is extremely important in design for stiffness. If this spacing is too short, system stiffness will be inadequate no matter how great the bearing stiffness. A major cause for failure in prismatic joints is surface fatigue wear (brinelling) of the ways caused by excessive ball loading due to high preload, moment loads and shock loads.

The large exposed precision surfaces in most prismatic joints make them much more sensitive than revolute joints to contamination, improper handling and environmental effects. They are also significantly more difficult to manufacture, properly assemble, and align.

Common types of sliding elements for prismatic motion are bronze or thermoplastic impregnated bushings. These bushings have the advantage of being low in cost, of having relatively high load capacity, and of working with unhardened or superficially hardened (i. e., plated or anodized) surfaces. Because the local or contact stress on the moving element is distributed and is low this element may be made of thin tube or an extruded shape. Another type of bushing in common use is the ball bushing. Ball bushings have the advantages of lower friction and greater precision than plain bushings. However, they require that the contacting surface of the joint be heat treated or hardened (generally to Rc 55 or greater) and of sufficient case and wall thickness to support the ball point contact loads and resulting high stresses.

Ball and roller slide assemblies are commonly used in robot prismatic joints. There are two basic categories of these slides; recirculating and non-recirculating. Non-recirculating ball and roller slides are used primarily for short-travel applications. They feature high precision and very low friction at the expense of being more sensitive to shock and relatively poor at accommodating moment loading. Recirculating ball slides are somewhat less precise but can carry higher loads than non-recirculating ball slides. They can also be set up to handle relatively large moment loads. Travel range can be up to several meters. Commercial recirculating ball slides and ways have greatly simplified the design and construction of linear axes, particularly in gantry and track mounted manipulators.

Another common type of prismatic robot joint is made up of cam followers, rollers, or wheels rolling on extruded, drawn, machined, or ground surfaces. In high-load applications the surfaces must be hardened before they are finish ground. Cam followers can be purchased



Fig. 4.13 RobotWorld is an integrated workcell with multiple robot modules that move on air bearings

with eccentric mounting shafts to facilitate setup and adjustment. Elastomer rollers provide quiet, smooth operation.

Two less common types of linear or prismatic joints feature flexures and air bearings. Flexure-based joints, whose motion results from elastic bending deformations of beam support elements, are used primarily for small, high-resolution, quasilinear motions. Air bearings require smooth surfaces and close control of tolerances as well as a constant supply of filtered, oil-free compressed air. Two- and three-degree-of-freedom air bearings (x, y, θ) can enable multiaxis motion with few moving parts (Fig. 4.13).

Joint Travel

For revolute joint configurations, the shoulder and elbow joints and links determine the gross volume of the work envelope (reachable workspace) of a robot manipulator arm. The wrist joints generally determine the orientation range (dexterous workspace) about a point within this work envelope. Larger joint travel may increase the number of possible manipulator configurations that will reach a particular location (increased task space). Wrist joint travel in excess of 360° and up to 720° can be useful for situations requiring controlled-path (e.g., straight-line) motion, synchronized motion such as small part assembly on a moving conveyor belt, or sensor-modified motions as in using machine vision to select and guide picking jumbled items from a bin. Continuous last-joint rotation is desirable in certain cases like loading or unloading a rotating machine or mating threaded parts.



Fig. 4.14 Wrist for surgery has no singularities within range of motion. Features tungsten cable actuation (after [4.49])

Additional joints and links, sometimes in the robot but more often in the end-effector, and specialized tooling also serve to increase the task space of a robot. Continuous and controlled-path robot motion requires planning to avoid singularities (regions where two or more joints may become aligned or nearly aligned) and the resulting unstable end-effector motion in these regions. Manipulator design coordinated with a well-planned workcell layout can improve the useful task space by placing critical motions well away from singularity regions. For example, a standard three-axis robot wrist has singularities 180° apart, which can be increased to 360° by implementing a somewhat more complex reduced singularity wrist. Such a wrist is used in a sheep-shearing robot to achieve multiple, long, continuous, smooth, constant-velocity, sensor-guided passes over the contoured body of a sheep to shear its wool [4.50, 51]. Figure 4.14 shows a wrist used for minimally-invasive surgery that has no singularities within its range of motion [4.49].

4.8 Actuators

Actuators supply the motive power for robots. Most robot actuators are commercially available components, which are adapted or modified, as necessary, for a specific robot application. Three commonly used actuator types are electromagnetic, hydraulic and pneumatic.

4.8.1 Electromagnetic Actuators

The most common types of actuators in robots today are electromagnetic actuators.

Electric Servomotors

Most robot manipulators use servomotors as a power source. Servomotors are designed to accurately follow the desired position, velocity and torque which change frequently and sometimes abruptly. They have structures similar to ordinary electric motors, but with low inertia and large torque capably for high accelerations. Typical servomotors used for robotic applications are permanent magnet (PM) DC motors and brushless DC (BLDC) motors.

Avidbots Ex1016
Page 125 of 2154

PM DC motors are widely used as a servomotor because of high torque, speed controllability over a wide range, well-behaved torque-speed characteristics, and adaptability to various types of control methods. The DC motor converts electrical energy into rotational or linear mechanical energy. It comes in many different types and configurations. The lowest-cost PM motors use ceramic (ferrite) magnets and robot toys and hobby robots often use this type of motor. A PM motor with a rare-earth (neodymium-iron-boron), NEO magnet stator produces the most torque and power for its size.

Brushless motors, also called AC servomotors or brushless DC motors, are widely used in industrial robots (Figs. 4.15 and 4.16). They substitute magnetic or optical sensors and electronic switching circuitry for the graphite brushes and copper bar commutator used in brush-type DC motors, thus eliminating the friction, spark-ing, and wear of commutating parts. Brushless motors generally have good performance at low cost because of the decreased complexity of the motor. However, the controllers for these motors are more complex and expensive than brush-type motor controllers. A passive multi-pole neodymium magnet rotor and a wire-wound iron stator of a brushless motor provide good heat dissipation and excellent reliability. Linear brushless motors function like unrolled rotary motors. They typically have a long, heavy, multiple magnet passive stator and a short, lightweight, electronically commutated wire-wound forcer (slider).



Fig. 4.15 The Baldor AC servomotor



Fig. 4.16 The Anorad brushless linear motor

Modeling of DC Motors and BLDC Motors. Both DC and BLDC motors can be described by almost identical equations although they have different structures, *de Silva* [4.52]. Fig. 4.17. The motor torque τ_m generated at the motor is given by,

$$\tau_m = K_t i_a \,, \tag{4.20}$$

where K_t is the torque constant [N m/A] and i_a is the armature current [A].

When the rotor is rotating, the back electromotive force (emf) v_b is induced as follows,

$$v_b = K_b \omega_m \,, \tag{4.21}$$

where K_b is the back emf constant [V/(rad/s)] and ω_m is the angular speed of the motor [rad/s].

The circuit equation is given by

$$v_a = R_a\, i_a + L_a \frac{\mathrm{d}i_a}{\mathrm{d}t} + v_b \,, \tag{4.22}$$

where v_a is the armature voltage (supply voltage to the armature) [V], R_a and L_a are the resistance [Ω] and inductance [H] of the armature winding, respectively.

Lastly, the mechanical equation is described by

$$J\frac{\mathrm{d}\omega_m}{\mathrm{d}t} + B\omega_m = \tau_m - \frac{\tau_l}{r} \,, \tag{4.23}$$

where J is the equivalent moment of inertia referred to the motor shaft [kg m^2], B is the equivalent viscous-friction coefficient referred to the motor shaft [N m/rad/s], τ_l is the load torque [N m], and r is the gear ratio. The quantities J and B are given by

$$J = J_m + \frac{J_l}{r^2}, \quad B = B_m + \frac{B_l}{r^2} \,, \tag{4.24}$$

where J_m and B_m are related to the motor including the gear connected to the motor shaft and J_l and B_l are referred to the load including the gear connected to the load shaft.

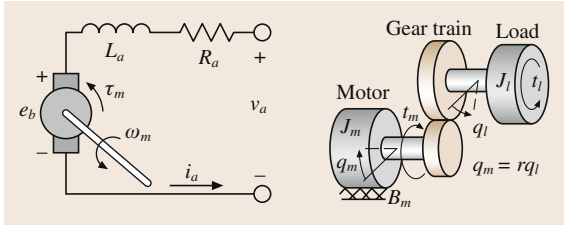


Fig. 4.17 Servomotor models

Avidbots Ex1016
Page 126 of 2154

The four equations above can be represented as the block diagram in Fig. 4.18.

In the servomotors, the electrical time constant $\tau_e = L_a/R_a$ and mechanical time constant $\tau_m = J/B$ have effects on their performance. The electrical time constant is much smaller than mechanical time constant because of the armature inductance is usually negligible. Considering this factor, the angular speed of the motor is described in terms of the armature voltage v_a and load torque τ_l as follows

$$\omega_m(s) = \frac{1}{JR_a\, s + (R_a\, B + K_t K_b)} \left(K_t v_a(s) - R_a \frac{\tau_l(s)}{r} \right). \tag{4.25}$$

The above equation is frequently used in simulations of both DC motor and BLDC motors.

Stepper Motors. Small, simple robots, such as benchtop adhesive dispensing robots, frequently use stepper or pulse motors of the permanent magnet (PM) hybrid type or sometimes the variable reluctance (VR) type (Fig. 4.19). These robots use open-loop position and velocity control. They are relatively low in cost and interface easily to electronic drive circuits. Microstep control can produce 10 000 or more discrete robot joint positions. In open-loop step mode the motors and robot motions have a significant settling time, which can be damped either mechanically or through the application of control algorithms. Power-to-weight ratios are lower for stepper motors than for other types of electric motors. Stepper motors operated with closed-loop control function similarly to direct-current (DC) or alternating-current (AC) servomotors (Fig. 4.20).

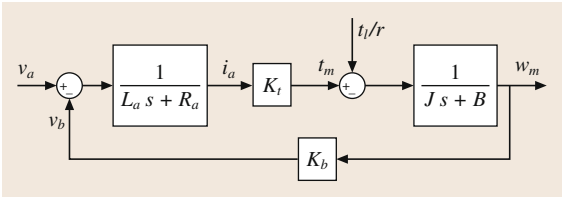


Fig. 4.18 Block diagram for servomotors

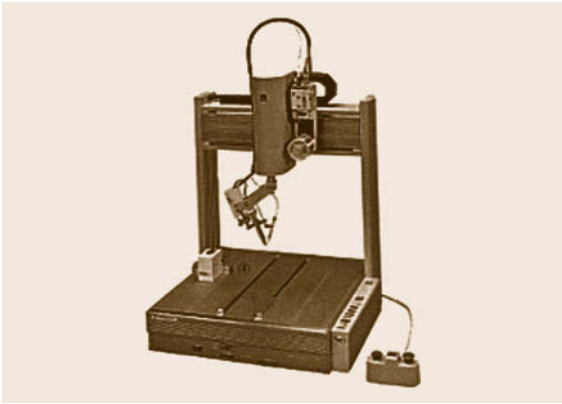


Fig. 4.19 The Sony robot uses open-loop permanent-magnet stepper motors

Permanent-Magnet DC Motor. The permanent-magnet, direct-current, brush-commutated motor is widely available and comes in many different types and configurations. The lowest-cost permanent-magnet motors use ceramic (ferrite) magnets. Robot toys and hobby robots often use this type of motor. Neodymium (NEO) magnet motors have the highest energy-product magnets, and in general produce the most torque and power for their size.

Ironless rotor motors, often used in small robots, typically have copper wire conductors molded into epoxy or composite cup or disk rotor structures. The advantages of these motors include low inductance, low friction, and no cogging torque. Disk armature motors have several advantages. They have short overall lengths, and because their rotors have many commutation segments, they produce a smooth output with low torque ripple. A disadvantage of ironless armature motors is that they have a low thermal capacity due to low mass and limited thermal paths to their case. As a result, when driven at high power levels they have rigid duty-cycle limitations or require forced-air cooling.

4.8.2 Hydraulic Actuators

Hydraulic actuators, chosen as power sources for the earliest industrial robots, offer very large force capability and high power-to-weight ratios. They convert hydraulic power into useful mechanical energy. The mechanical motion produced may be linear for hy-



Fig. 4.20 The Adept robot uses closed-loop control and variable-reluctance motors (VIDEO 644)

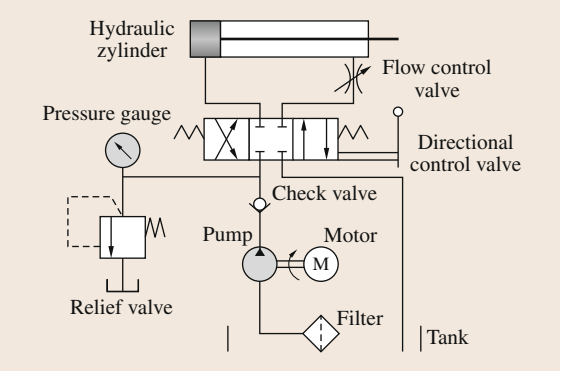


Fig. 4.21 General hydraulic circuit

draulic cylinders or rotary for hydraulic motors and vane actuators.

Figure 4.21 shows the basic hydraulic circuit composed of oil tank, hydraulic pump, check valve, relief valve, directional control valve and hydraulic actuator. Hydraulic fluid pressurized by the hydraulic pump is conveyed to the hydraulic machine to perform the specified task. To this end, the pressure, flow, and direction of hydraulic fluid is changed by the hydraulic control valves such as a pressure control valve, flow control valve, and directional control valve, respectively. The directional control valve changes the flow direction of the hydraulic fluid which is pressurized in the hydraulic pump and thus the direction of actuator motion. The flow control valve changes the flow rate of hydraulic fluid flow and thus the speed of the hydraulic actuator. For safety, the highest pressure is limited by the pressure control valve such as a relief valve.

Hydraulic actuators have several advantages and disadvantages due to the use of high pressure fluid. They can offer very large force or torque and high power-to-weight ratios. Both linear and rotary motions are readily available with small inertia of the moving part. However, the hydraulic power supply is bulky and the cost of the proportional, fast-response servovalves are high. Leaks and maintenance issues have limited the use and applica-tion of hydraulically powered robots.

There have been many applications of hydraulic actuators to robotics. For high forces or torques and speeds, hydraulic servo actuators out-perform current electromagnetic actuators. With proper design, leakage can be virtually eliminated, *Hollerbach* et al. [4.53]. Figure 4.22 shows some typical applications of hydraulic actuators, [4.54–56].

4.8.3 Pneumatic Actuators

Pneumatic actuators are similar to hydraulic actuators. Pneumatic actuators convert energy (in the form of compressed air) into mechanical motion, which may be linear or rotary. Pneumatic actuators are primarily found in simple manipulators. Typically they provide uncontrolled motion between mechanical limit stops. These actuators provide good performance in point-to-point motion. They are simple to control and are low in cost. Pneumatic motors have several advantages over electric motors. They are relatively safe in the explosive environment. They are also less affected by ambient temperature and humidity than electric motors. Although a few small actuators may be run with typical factory air supplies, extensive use of pneumatic-actuated robots requires the purchase and installation of a costly dedicated compressed-air source. Pneumatic actuators have low energy efficiency.

Pneumatic systems consist of pneumatic generator, pneumatic valves, pneumatic actuator and pipes. A pneumatic generator produces compressed air using an air compressor. Pneumatic valves are used to control the pressure, flow rate and direction. The mechanical motion produced may be linear for pneumatic cylinders or rotary for pneumatic motors.

Pneumatic actuators are not used for applications requiring large forces or torques since they produce less power than hydraulic actuators or electric actuators. However, they are used in robot hands or artificial muscles, which require high power-to-weight ratios. Pneumatic artificial muscles are contractile or extensional devices operated by pressurized air filling a pneumatic bladder. They are usually grouped in the agonist

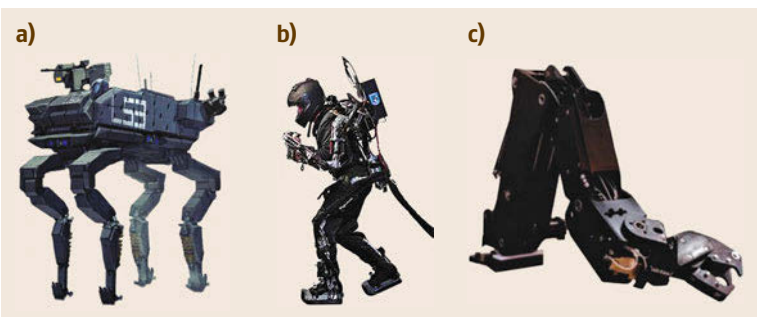


Fig.4.22a–c Applications of hydraulic actuators to robot: **(a)** BigDog (Boston dynamics) (VIDEO 645), **(b)** Sarcos exoskeleton (VIDEO 646) (Raytheon) **(c)** Magnum 7 (International Submarine Engineering)

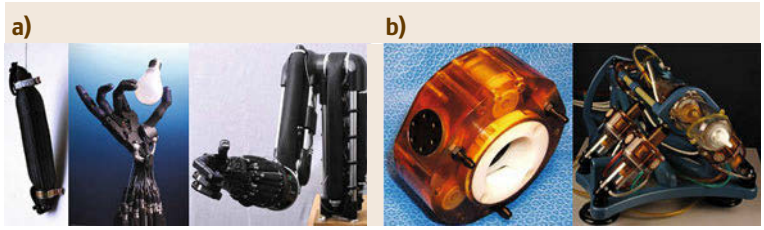


Fig.4.23a,b Applications of pneumatic actuator: **(a)** robot hand and arm with artificial muscle (Shadow robot) and **(b)** pneumatic step motor and MrBot (Urobotics, Johns Hopkins)

and antagonist pairs. Furthermore, they can be used for medical robots since they are not affected by magnetic field, and for robots in explosive environments because there is no electrical arcing that exists in electromagnetic actuators. Figure 4.23 shows some robotic applications of pneumatic actuators [4.57, 58].

4.8.4 Other Actuators

A wide variety of other types of actuators have been applied to robots. A sampling of these include, thermal, shape-memory alloy (SMA), bimetallic, chemical, piezoelectric, magnetostrictive, electroactive polymer (EAP), bladder, and micro-electromechanical system (MEMS) actuators (see Figs. 4.24 and 4.25). Most of these actuators have been applied to research and special application robots rather than volume production industrial robots. An example of a piezoelectric actuator powered robot is the six-axis PI (Physik Instrumente) piezo hexapod with sub-nanometer resolution shown in Fig. 4.26.



Fig. 4.24 The artificial muscle EAP motor

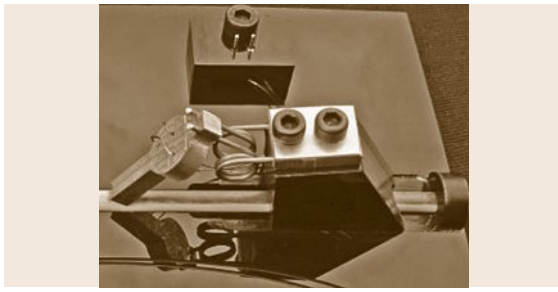


Fig. 4.25 The Elliptec piezoelectric motor

4.8.5 Transmissions

The purpose of a transmission or drive mechanism is to transfer mechanical power from a source to a load. The design and selection of a robot drive mechanism requires consideration of motion, load, and power requirements and the placement of the actuator with respect to the joint. The primary considerations in transmission design are stiffness, efficiency, and cost. Backlash and windup impact drive stiffness especially in robot applications where motion is constantly reversing and loading is highly variable. High transmission stiffness and low or no backlash result in increased friction losses. Most robot transmission elements have good efficiencies when they are operating at or near their rated power levels but not necessarily when lightly loaded. Larger than necessary drives add weight, inertia and friction loss to the system. Underdesigned drives have lower stiffness, can wear rapidly in continuous or in high duty cycle operation or fail due to accidental overloads.

Joint actuation in robots is generally performed by drive mechanisms which interface the actuator (mechanical work source) to the robot links through the joints in an energy-efficient manner. A variety of drive mechanisms are incorporated in practical robots. The transmission ratio of the drive mechanism sets the torque, speed, and inertia relationship of the actuator to the link. Proper placement, sizing, and design of the drive mechanisms set the stiffness, mass, and overall operational performance of the robot. Most modern robots incorporate efficient, overload damage resistant, back-driveable drives.



Fig. 4.26 A six-axis Physik Instrumente (PI) piezo hexapod with sub-nanometer resolution (VIDEO 648)

Avidbots Ex1016
Page 129 of 2154

Direct Drives

The direct drive is kinematically the simplest drive mechanism. In the case of pneumatic or hydraulic actuated robots, the actuator is directly connected between the links. Electric direct-drive robots employ high-torque, low-speed rotary or linear motors directly interfaced to the links. The complete elimination of free play and smooth torque transmission are features of a direct drive. However, there is often a poor dynamic (inertia ratio) match of the actuator to the link requiring a larger, less energy efficient, actuator.

Band Drives

A variant of direct drive is band drive. A thin alloy steel or titanium band is fixed between the actuator shaft and the driven link to produce limited rotary or linear motion. Drive ratios in the order of up to 10 : 1 (10 actuator revolutions for 1 revolution of the joint) can be obtained. Actuator mass is also moved away from the joint – usually toward the base, to reduce robot inertia and gravity loading. It is a smoother and generally stiffer drive than a cable or belt drive.

Belt Drives

Synchronous (toothed) belts are often employed in drive mechanisms of smaller robots and some axes of larger robots. These function much the same as band drives, but have the ability to drive continuously. Multiple stages (two or three) are occasionally used to produce large drive ratios (up to 100 : 1). Belt tension is controlled with idlers or center adjustment. The elasticity and mass of long belts can cause drive instability and thus increased robot settling time.

Gear Drives

Spur or helical gear drives provide reliable, sealed, low-maintenance power transmission in robots. They are used in robot wrists where multiple axes intersect and compact drive arrangements are required. Large-diameter gears are used in the base joints of larger robots to handle high torques with high stiffness. Gears are often used in stages and often with long drive shafts, enabling large physical separation between actuator and driven joint. For example, the actuator and one stage of reduction may be located near the elbow driving another stage of gearing or differential in a wrist through a long hollow drive shaft (Fig. 4.1).

Planetary gear drives are often integrated into compact gearmotors (Fig. 4.27). Minimizing backlash (free play) in a joint gear drive requires careful design, high-precision and rigid support to produce a drive mechanism which does not sacrifice stiffness, efficiency and accuracy for low backlash. Backlash in robots is controlled by a number of methods including selective assembly, gear center adjustment, and proprietary anti-backlash designs.

Worm Gear Drives

Worm gear drives are occasionally used in low-speed robot manipulator applications. They feature right-angle and offset drive capability, high ratios, simplicity, good stiffness and load capacity. They have poor efficiency, which makes them non-back-driveable at high ratios. This causes the joints to hold their position when unpowered but also makes them prone to damage by attempts to manually reposition the robot.

Proprietary Drives

Proprietary drives are widely used in standard industrial manipulators. The harmonic drive and the rotary vector (RV) drive are two examples of compact, low-backlash, high-torque-capability drives using special gears, cams, and bearings (Figs. 4.28 and 4.29).

Harmonic drives are frequently used in very small to medium-sized robots. These drives have low backlash, but its flexspline allows elastic windup and low stiffness during small reversing movements. RV drives are

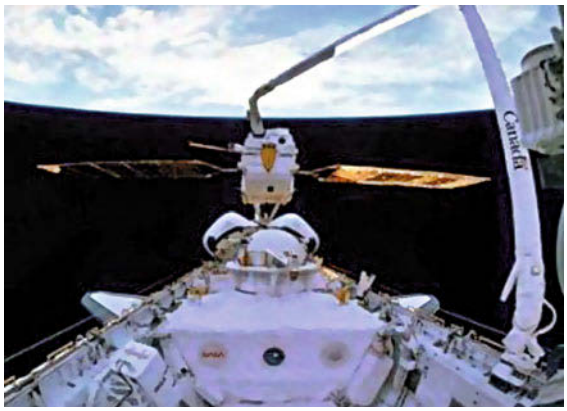


Fig. 4.27 The Space Shuttle robot arm has planetary gear joint drives



Fig. 4.28 The harmonic drive (VIDEO 649)

usually used in larger robots, especially those subject to overloads and shock loading.

Linear Drives

Direct-drive linear actuators incorporate a linear motor with a linkage to a linear axis. This linkage is often merely a rigid or flexure connection between the actuator forcer and the robot link. Alternatively, a packaged linear motor with its own guideways is mechanically connected directly to a linear axis. Direct linear electromagnetic drives feature zero backlash, high stiffness, high speeds, and excellent performance but are heavy, have poor energy efficiency, and cost more than other types of linear drives.

Ball Screws

Ball–screw-based linear drives efficiently and smoothly convert rotary actuator motion into linear motion. Typically, a recirculating ball nut mates with a ground and hardened alloy steel screw to convert rotary motion into linear motion. Ball screws can be easily integrated into linear axes. Compact actuator/drive packages are available, as well as components for custom integration. Stiffness is good for short and medium travel, however it is lower for long motions because the screw can only be supported at its ends. Low or zero backlash can be obtained with precision-ground screws. Speeds are limited by screw dynamic stability so rotating nuts enable higher speeds. Low-cost robots may employ plain screw drives featuring thermoplastic nuts on smooth rolled thread screws.



Fig. 4.29 The Nabtesco RV drive

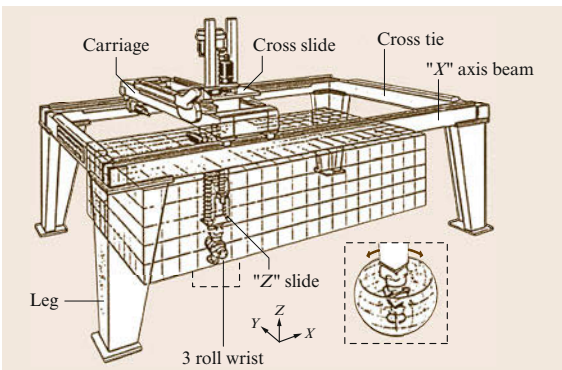


Fig. 4.30 The NASA gantry robot

Rack-and-Pinion Drives

These traditional components are useful for long motions where the guideways are straight or even curved. Stiffness is determined by the gear/rack interface and independent of length of travel. Backlash can be diffi-

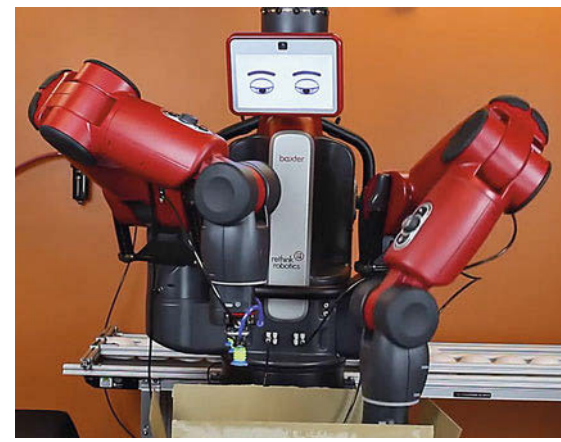


Fig. 4.31 Two-arm collaborative robot with Series Elastic Actuators (after [4.59])



Fig. 4.32 Series Elastic Actuator assembly in knee-ankle-foot gait rehabilitation robot (after [4.60])

cult to control as rack-to-pinion center tolerances must be held over the entire length of travel. Dual pinion drives are sometimes employed to deal with backlash by providing active preload. Forces are generally lower than with screws due to lower ratios. Small-diameter (low teeth count) pinions have poor contact ratios, resulting in vibration. Sliding involute tooth contact requires lubrication to minimize wear. These catalog stock drive components are often used on large gantry robots and track-mounted manipulators (Fig. 4.30).

Compliant Actuators. Compliance in the form of elasticity in a drive can be an asset or a defect in a robot. High overall robot stiffness traditionally produces fast response, high positioning accuracy and simplified control. However the contact and interaction forces developed due to unexpected errors in dealing with the workpieces, tools, workplace and the environment can cause damage to the robot, the surroundings, and injury to persons. By adding controlled and measureable compliance to actuators, the elasticity of the robot can be usefully increased.

One class of reduced stiffness actuator is the Series Elastic Actuator (SEA). It features an elastic output element (spring) with a displacement sensor (measures spring stretch) in series with a stiff actuator and transmission component. With an appropriate controller, the stiff classical position control actuator acts as a force actuator ($F = -kx$), effectively isolating the drive inertia from the load inertia. This limits the forces resulting from collisions or forced compliance often encountered when working in unstructured environments and around persons. SEAs and similar compliant actuators and compliant actuator configurations are discussed in Chap. 69. Collaborative robots are designed to safely work next to or in contact with persons, Fig. 4.31 [4.59], and are an example of robots featuring compliant actuators. Figure 4.32 shows compliant actuators used in an exoskeleton for gait rehabilitation [4.60].

Other Drive Components

Splined shafts, kinematic linkages (four-bar, slider-crank mechanisms, etc.) chains, cables, flex couplings, clutches, brakes, and limit stops are some examples of other mechanical components used in robot drive mechanisms (Fig. 4.8). The Yaskawa RobotWorld assembly and process automation robot system features a magnetically suspended direct electromagnetic drive planar (2 axis) motor with no internal moving parts floating on a virtually frictionless planar air bearing. (Fig. 4.12).

4.9 Robot Performance

Industrial robot performance is often specified in terms of functional operations and cycle time. For assembly robots the specification is often the number of typical pick-and-place cycles per minute. Arc-welding robots are specified with a slow weld pattern and weave speed as well as by a fast repositioning speed. For painting robots, the deposition or coverage rate and spray pattern speed are important. Peak robot velocity and acceleration catalog data are generally just calculated numbers and will vary due to dynamic (inertia) and static (gravity) coupling between robot joints due to configuration changes as a robot moves.

4.9.1 Robot Speed

Maximum joint velocity (angular or linear) is not an independent value. For longer motions it is often limited by servomotor bus voltage or maximum allowable motor speed. For manipulators with high accelerations, even short point-to-point motions may be velocity limited. For low-acceleration robots, only gross motions will be velocity limited. Typical peak end-effector speeds can range up to 20 m/s for large or fast manipulators.

4.9.2 Robot Acceleration

In most modern manipulators, because the payload mass is small when compared with the manipulator mass, more power is spent accelerating the manipulator than the load. Acceleration affects gross motion time as well as cycle time (gross motion time plus settling time). Manipulators capable of greater acceleration tend to be stiffer manipulators. In high-performance robot manipulators, acceleration and settling time are more important design parameters than velocity or load capacity. Maximum acceleration for some assembly and material handling robots is in excess of 10 g with light payloads.

4.9.3 Repeatability

This specification represents the ability of the manipulator to return repeatedly to the same location. Depending on the method of teaching or programming the manipulator, most manufacturers intend this figure to indicate the radius of a sphere enclosing the set of locations to which the arm returns when sent from the same origin by the same program with the

same load and setup conditions. This sphere may not include the target point because calculation round-off errors, simplified calibration, precision limitations, and differences during the teaching and execution modes can cause significantly larger errors than those just due to friction, unresolved joint and drive backlash, servo system gain, and structural and mechanical assembly clearances and play. The designer must seriously consider the real meaning of the required repeatability specification. Repeatability is important when performing repetitive tasks such as blind assembly or machine loading. Typical repeatability specifications range from 1–2 mm for large spot-welding robots to 0.005 mm (5 μm) for precise micropositioning manipulators.

4.9.4 Resolution

This specification represents the smallest incremental motion that can be produced by the manipulator. Resolution is important in sensor-controlled robot motion and in fine positioning. Although most manufacturers calculate system resolution from the resolution of the joint position encoders, or from servomotor and drive step size, this calculation is misleading because system friction, windup, backlash, and kinematic configuration affect the system resolution. Typical encoder or resolver resolution is 10^{14}–10^{25} counts for full-axis or joint travel, but actual physical resolution may be in the range 0.001–0.5 mm. The useful resolution of a multijoint serial-link manipulator is worse than that of its individual joints.

4.9.5 Accuracy

This specification covers the ability of a robot to position its end-effector at a preprogrammed location in space. Robot accuracy is important in the performance of nonrepetitive types of tasks programmed from a database, or for taught tasks that have been remapped or offset owing to measured changes in the installation.

Accuracy is a function of the precision of the arm kinematic model (joint type, link lengths, angles between joints, any accounting for link or joint deflections under load, etc.), the precision of the world, tool, and fixture models, and the completeness and accuracy of the arm solution routine. Although most higher-level robot programming languages support arm solutions, these solutions usually model only simplified rigid-body kinematic configurations. Thus, manipulator accuracy becomes a matter of matching the robot geometry to the robot solution in use by precisely measuring and calibrating link lengths, joint angles, and mounting positions.

Typical accuracies for industrial manipulators range from ±10 mm for uncalibrated manipulators that have poor computer models to ±0.01 mm for machine-tool-like manipulators that have controllers with accurate kinematic models and solutions and precisely manufactured and measured kinematic elements.

4.9.6 Component Life and Duty Cycle

The three subassemblies in an electrically powered robot with the greatest failure problems are the actuators (servomotors), transmissions, and power and signal cables. Mean time between failures (MTBF) should be a minimum of 5000 h on line, and ideally at least 10 000 operating hours should pass between major component preventive maintenance replacement schedules.

Worst-case motion cycles must be assumed as most current robot installations are used in generally repetitive tasks. Small-motion design-cycle life (less than 5% of joint travel range) for assembly robots should be 20–100 million full bidirectional cycles. Large-motion cycle life (greater than 50% of full joint range) should typically be 5–40 million cycles.

Short-term peak performance is frequently limited by maximum drive loading, whereas long-term, continuous, performance is limited by motor heating. Rather than design for equal levels of short- and long-term performance, cost savings and performance improvements can result from designing for an anticipated duty cycle. This allows the use of smaller, lower-inertia, lighter motors. Industrial robots usually become obsolete and are replaced before they reach their design cycle life.

4.9.7 Collisions

In the course of operation, unforeseen or unexpected situations may occasionally result in a collision involving the manipulator, its tools, the workpiece, or other objects in the workplace. These accidents may result in no, little, or extensive damage, depending in large part on the design of the manipulator. Crash-resistant design options should be considered early in the design process if the time lost or cost of such accidents could be significant. Typical damage due to accidents include fracture or shear failures of gear teeth or shafts, dented or bent link structures, slipping of gears or pulleys on shafts, cut or severely abraded or deformed wires, cables or hoses, and broken connectors, fittings, limit stops or switches. Compliant elements such as overload (slip) clutches, elastic members, and padded surfaces can be incorporated to reduce shock loads and help decouple or isolate the actuators and drive components in the event of such collisions.

Avidbots Ex1016
Page 133 of 2154

4.10 Conclusions and Further Reading

The mechanical design of a robot is an iterative process involving engineering, technical, and application-specific considerations evaluations, and choices. The final design should reflect consideration of detailed task requirements rather than simply broad specifications. Proper identification and understanding of these requirements is a key to achieving the design goals.

Design and choice of specific components involves tradeoffs. A purely static, rigid-body approach to manipulator design is often used, but is not always sufficient. Mechanical system stiffness, natural frequencies, control system compatibility, and intended robot applications and installation requirements must be considered.

There are many opportunities for further reading on the design of the mechanisms and actuation that form the core of a robotic system. A well-known and useful reference for robot design is *Rivin* [4.44].

Craig [4.6] and *Tsai* [4.9] provide the mathematical relations between the mechanical structure of a robot and its workspace and mechanical advantage. *Sclater* and *Chironis* [4.48] is a reprint of a valuable compendium of devices useful for a variety of applications, such as joint drives and transmissions. See *McCarthy* and *Soh* [4.34] for geometric techniques to design specialized mechanisms.

Juvinall and *Marshek* [4.45] and *Shigley* and *Mischke* [4.46, 47] are important references for the design of the components such as the link structure, bearings, and transmissions that are central to the effective mechanical performance of robotic systems.

Although many design decisions can be made through the application of straightforward algorithms and equations, a multitude of other important considerations transform the challenge of robot design into one requiring good engineering judgment.

Video-References

VIDEO 640 A parallel robot
available from http://handbookofrobotics.org/view-chapter/04/videodetails/640

VIDEO 642 Three-fingered robot hand
available from http://handbookofrobotics.org/view-chapter/04/videodetails/642

VIDEO 643 Robotics milking system
available from http://handbookofrobotics.org/view-chapter/04/videodetails/643

VIDEO 644 SCARA robots
available from http://handbookofrobotics.org/view-chapter/04/videodetails/644

VIDEO 645 Big Dog –Applications of hydraulic actuators
available from http://handbookofrobotics.org/view-chapter/04/videodetails/645

VIDEO 646 Raytheon Sarcos exoskeleton
available from http://handbookofrobotics.org/view-chapter/04/videodetails/646

VIDEO 648 PI piezo hexapod
available from http://handbookofrobotics.org/view-chapter/04/videodetails/648

VIDEO 649 Harmonic drive
available from http://handbookofrobotics.org/view-chapter/04/videodetails/649

References

4.1 D.T. Greenwood: *Classical Dynamics* (Prentice Hall, Upper Saddle River 1977)
4.2 F.C. Moon: *Applied Dynamics* (Wiley, New York 1998)
4.3 S.-M. Song, K.J. Waldron: *Machines that Walk: The Adaptive Suspension Vehicle* (MIT Press, Cambridge 1988)
4.4 M.T. Mason, J.K. Salisbury: *Robot Hands and the Mechanics of Manipulation* (MIT Press, Cambridge 1985)
4.5 R.P. Paul: *Robot Manipulators: Mathematics, Programming, and Control* (MIT Press, Cambridge 1981)
4.6 J.J. Craig: *Introduction to Robotics: Mechanics and Control* (Addison-Wesley, Reading 1989)
4.7 O. Bottema, B. Roth: *Theoretical Kinematics* (North-Holland, Amsterdam 1979)
4.8 J.M. McCarthy: *An Introduction to Theoretical Kinematics* (MIT Press, Cambridge 2013)
4.9 L.W. Tsai: *Robot Analysis. The Mechanics of Serial and Parallel Manipulators* (Wiley, New York 1999)
4.10 T. Lozano-Perez: Spatial Planning: A configuration space approach, IEEE Trans. Comput. **32**(2), 108–120 (1983)
4.11 J.C. Latombe: *Robot Motion Planning* (Kluwer, Boston 1991)
4.12 R. Vijaykumar, K. Waldron, M.J. Tsai: Geometric optimization of manipulator structures for working vol-

Avidbots Ex1016
Page 134 of 2154

ume and dexterity. In: *Kinematics of Robot Manipulators*, ed. by J.M. McCarthy (MIT Press, Cambridge 1987) pp. 99–111
4.13 K. Gupta: On the nature of robot workspace. In: *Kinematics of Robot Manipulators*, ed. by J.M. McCarthy (MIT Press, Cambridge 1987) pp. 120–129
4.14 I. Chen, J. Burdick: Determining task optimal modular robot assembly configurations, Proc. IEEE Robotics Autom. Conf. (1995) pp. 132–137
4.15 P. Chedmail, E. Ramstei: Robot mechanisms synthesis and genetic algorithms, Proc. IEEE Robotics Autom. Conf. (1996) pp. 3466–3471
4.16 P. Chedmail: Optimization of multi-DOF mechanisms. In: *Computational Methods in Mechanical Systems*, ed. by J. Angeles, E. Zakhariev (Springer, Berlin, Heidlberg 1998) pp. 97–129
4.17 C. Leger, J. Bares: Automated Synthesis and Optimization of Robot Configurations, Proc. ASME Design Tech. Conf., Atlanta (1998), paper no. DETC98/Mech-5945 CD-ROM
4.18 F.C. Park: Distance metrics on the rigid body motions with applications to mechanism design, ASME J. Mech. Des. **117**(1), 48–54 (1995)
4.19 J.M.R. Martinez, J. Duffy: On the metrics of rigid body displacements for infinite and finite bodies, ASME J. Mech. Des. **117**(1), 41–47 (1995)
4.20 M. Zefran, V. Kumar, C. Croke: Choice of Riemannian metrics for rigid body kinematics, Proc. ASME Design Tech. Conf., Irvine (1996), paper no. DETC96/Mech-1148
4.21 Q. Lin, J.W. Burdick: On well-defined kinematic metric functions, Proc. Int. Conf. Robotics Autom., San Francisco (2000) pp. 170–177
4.22 C. Gosseli: On the design of efficient parallel mechanisms. In: *Computational Methods in Mechanical Systems*, ed. by J. Angeles, E. Zakhariev (Springer, Berlin, Heidelberg 1998) pp. 68–96
4.23 J.V. Albro, G.A. Sohl, J.E. Bobrow, F. Park: On the computation of optimal high-dives, Proc. Int. Conf. Robotics Autom., San Francisco (2000) pp. 3959–3964
4.24 G.E. Shilov: *An Introduction to the Theory of Linear Spaces* (Dover, New York 1974)
4.25 J.K. Salisbury, J.J. Craig: Articulated hands: Force control and kinematic issues, Int. J. Robotics Res. **1**(1), 4–17 (1982)
4.26 J. Angeles, C.S. Lopez-Cajun: Kinematic isotropy and the conditioning index of serial manipulators, Int. J. Robotics Res. **11**(6), 560–571 (1992)
4.27 J. Angeles, D. Chabla: On isotropic sets of points in the plane. Application to the design of robot architectures. In: *Advances in Robot Kinematics*, ed. by J. Lenarčič, M.M. Stanišić (Kluwer, Boston 2000) pp. 73–82
4.28 E.F. Fichter: A Stewart platform-based manipulator: General theory and practical construction. In: *Kinematics of Robot Manipulators*, ed. by J.M. McCarthy (MIT Press, Cambridge 1987) pp. 165–190
4.29 J.P. Merlet: *Parallel Robots* (Kluwer, Boston 1999)
4.30 J.M. Hervé: Analyse structurelle des méchanismes par groupe des déplacements, Mech. Mach. Theory **13**(4), 437–450 (1978)
4.31 J.M. Hervé: The Lie group of rigid body displacements, a fundamental tool for mechanism design, Mech. Mach. Theory **34**, 719–730 (1999)
4.32 A.P. Murray, F. Pierrot, P. Dauchez, J.M. McCarthy: A planar quaternion approach to the kinematic synthesis of a parallel manipulator, Robotica **15**(4), 361–365 (1997)
4.33 A. Murray, M. Hanchak: Kinematic synthesis of planar platforms with RPR, PRR, and RRR chains. In: *Advances in Robot Kinematics*, ed. by J. Lenarčič, M.M. Stanišić (Kluwer, Boston 2000) pp. 119–126
4.34 J.M. McCarthy, G.S. Soh: *Geometric Design of Linkages*, 2nd edn. (Springer, Berlin, Heidelberg 2010)
4.35 V. Kumar: Instantaneous kinematics of parallel-chain robotic mechanisms, J. Mech. Des. **114**(3), 349–358 (1992)
4.36 C. Gosselin, J. Angeles: The optimum kinematic design of a planar three-degree-of-freedom parallel manipulator, ASME J. Mech. Transmiss. Autom. Des. **110**(3), 35–41 (1988)
4.37 J. Lee, J. Duffy, M. Keler: The optimum quality index for the stability of in-parallel planar platform devices, Proc. ASME Design Eng. Tech. Conf., Irvine (1996), paper no. 96-DETC/MECH-1135
4.38 J. Lee, J. Duffy, K. Hunt: A practical quality index based on the octahedral manipulator, Int. J. Robotics Res. **17**(10), 1081–1090 (1998)
4.39 S.E. Salcudean, L. Stocco: Isotropy and actuator optimization in haptic interface design, Proc. Int. Conf. Robotics Autom., San Francisco (2000) pp. 763–769
4.40 L.-W. Tsai, S. Joshi: Kinematics and optimization of a spatial 3-UPU parallel manipulator, J. Mech. Des. **122**, 439–446 (2000)
4.41 Q. Jin, T.-L. Yang: Theory for topology synthesis of parallel manipulators and its application to three-dimension-translation parallel manipulators, J. Mech. Des. **126**(3), 625–639 (2004)
4.42 X. Kong, C.M. Gosselin: Type synthesis of three-degree-of-freedom spherical parallel manipulators, Int. J. Robotics Res. **23**, 237–245 (2004)
4.43 T.A. Hess-Coelho: Topological synthesis of a parallel wrist mechanism, J. Mech. Des. **128**(1), 230–235 (2006)
4.44 E.I. Rivin: *Mechanical Design of Robots* (McGraw-Hill, New York 1988) p. 368
4.45 R.C. Juvinall, K.M. Marshek: *Fundamentals of Machine Component Design*, 4th edn. (Wiley, New York 2005) p. 832
4.46 J.E. Shigley, C.R. Mischke: *Mechanical Engineering Design*, 7th edn. (McGraw-Hill, New York 2004) p. 1056
4.47 J.E. Shigley, C.R. Mischke: *Standard Handbook of Machine Design*, 2nd edn. (McGraw-Hill, New York 1996) p. 1700
4.48 N. Sclater, N. Chironis: *Mechanisms and Mechanical Devices Sourcebook*, 4th edn. (McGraw Hill, New York 2007) p. 512
4.49 Intuitive Surgical EndoWrist Insturments, http://www.intuitivesurgical.com/products/instruments/
4.50 J.P. Trevelyan: Sensing and control for shearing robots, IEEE Trans. Robotics Autom. **5**(6), 716–727 (1989)

Avidbots Ex1016
Page 135 of 2154

4.51 J.P. Trevelyan, P.D. Kovesi, M. Ong, D. Elford: ET: A wrist mechanism without singular positions, Int. J. Robotics Res. **4**(4), 71–85 (1986)
4.52 C. de Silva: *Sensors and Actuators: Control Systems Instrumentation* (CRC, Boca Raton 2007)
4.53 J. Hollerbach, I. Hunter, J. Ballantyne: A Comparative analysis of actuator technologies for robotics, Robotics Rev. **2**, 299–342 (1992)
4.54 Boston Dynamics, Waltham, MA, USA: Big Dog Robot, http://www.bostondynamics.com/robot_bigdog.html
4.55 Raytheon, Waltham MA, USA: Sarcos Exoskeleton, http://www.popsci.com/category/tags/raytheon-sarcos-xos
4.56 International Submarine Engineering, Port Coquitlam, BC, Canada: Magnum 7, http://www.ise.bc.ca/manips.html
4.57 Shadow Robot Company, London, UK: Air muscles and pneumatic hands, http://www.shadowrobot.com/tag/muscle-hand
4.58 Johns Hopkins, Baltimore, USA: Pneumatic Stepper Motor and MrBot, http://urobotics.urology.jhu.edu/projects/PneuStep
4.59 The Baxter Robot Rethink Robotics, http://www.rethinkrobotics.com/products/baxter/
4.60 Singapore Institute for Neurotechnology, NeuroRehabilitation Laboratory: http://www.sinapseinstitute.org/projects/neurorehabilitation

Avidbots Ex1016
Page 136 of 2154

Multimedia Contents

5. Sensing and Estimation

Henrik I. Christensen, Gregory D. Hager

Sensing and estimation are essential aspects of the design of any robotic system. At a very basic level, the state of the robot itself must be estimated for feedback control. At a higher level, perception, which is defined here to be task-oriented interpretation of sensor data, allows the integration of sensor information across space and time to facilitate planning.

This chapter provides a brief overview of common sensing methods and estimation techniques that have found broad applicability in robotics. The presentation is structured according to a process model that includes sensing, feature extraction, data association, parameter estimation, and model integration. Several common sensing modalities are introduced and characterized. Common methods for estimation in linear and nonlinear systems are discussed, including statistical estimation, the Kalman filter, and sample-based methods. Strategies for robust estimation are also briefly described. Finally, several common representations for estimation are introduced.

5.1 **Introduction** 91
5.2 **The Perception Process** 92
5.3 **Sensors** 94
5.4 **Estimation Processes** 98
5.4.1 Point Estimation 98
5.4.2 Other Approaches to Estimation 101
5.4.3 Robust Estimation Methods 103
5.4.4 Data Association Techniques 105
5.4.5 Modeling Sensors 108
5.4.6 Other Uncertainty Management Methods 108
5.5 **Representations** 109
5.5.1 Raw Sensor Representations 109
5.5.2 Grid-Based Representations 109
5.5.3 Feature Representations 110
5.5.4 Symbolic/Graph-Based Models 110
5.6 **Conclusions and Further Readings** 111
References 111

5.1 Introduction

Controlling a robotic system would be relatively simple if a complete model of the environment was available, and if the robot actuators could execute motion commands perfectly relative to this model. Unfortunately, in most cases of interest, a complete world model is not available, and perfect control of mechanical structures is never a realistic assumption. Sensing and estimation are a means of compensating for this lack of complete information. Their role is to provide information about the state of the environment and the state of the robot system as a basis for control, decision making, and interaction with other agents in the environment, such as humans.

For the purposes of discussion, we will differentiate between sensing and estimation to recover the state of the robot itself, referred to as *proprioception*, versus sensing and estimation to recover the state of the external world, referred to as *exteroception*. In practice, most robot systems are designed to have the proprioception necessary to estimate and control their own physical state. On the other hand, recovering the state of the world from sensor data is usually a much larger and more complex problem.

Early work on computational perception for robotics assumed that one could recover a complete general-purpose model of the environment, use such

a model to make decisions, and subsequently act on them, as for example presented by [5.1]. More recently it has become apparent that such an approach is not realistic. Indeed, considering that sensor-based robots now appear in diverse applications such as mobile surveillance, high-performance manipulation, and medical interventions, it is clear that appropriate sensing and estimation for a given system must be highly task dependent. Consequently, the discussion here is organized along the lines of task-oriented sensing and estimation of the external world.

Sensing and estimation together can be viewed as the process of transforming a physical quantity into a computer representation that can be used for further processing. Sensing is thus closely tied to transducers that transform some physical entity into a signal that can be processed by a computer. Sensing is also intimately tied to perception, the process of representing the sensory information in an task-oriented model of the world. However, sensor data is usually corrupted in various ways that complicate this process. Statistical noise arises from the transducer, discretization is introduced in the digitization process, and ambiguity is introduced by poor sensor selectivity to name a few examples. Estimation methods are thus introduced to support appropriate integration of information into models of the environment and for improvement of the signal-to-noise ratio.

In this chapter the general characteristics of sensing and estimation are introduced, while more in-depth presentations of select topics are provided in Part C of the handbook. In Sect. 5.2 the overall sensing/perception process is introduced. In Sect. 5.3 different kinds of sensors are introduced and some key characteristics are presented. Estimation of world representations can utilize a number of different methods involving both parametric and nonparametric techniques as discussed in Sect. 5.4. For model-based integration a variety of different representations can be used, as described in Sect. 5.5.

5.2 The Perception Process

The input to the perception process is typically twofold: (1) digital data from a number of sensors/transducers, and (2) a partial model of the environment (a world model) that includes information about the state of the robot and other relevant entities in the external world. The sensor data itself can take on a number of different forms such as a scalar or vector value $\boldsymbol{x}(\alpha, \beta)$ acquired over a time series $x(t)$, a scan $x_t(\theta_i)$, a vector field $\boldsymbol{x}$ or a three-dimensional volume $x(\rho, \theta, \phi)$. In many cases, a system must integrate data from several disparate sensors, for example, an estimate of the position of a mobile robot may integrate data from axis encoders, vision, global positioning system (GPS) data, and inertial sensors.

To further structure the discussion in this chapter, we adopt a general model of the perception process as shown in Fig. 5.1. In this model, we have included the most common operations applied to integrate sensor data with a world model. Depending on the task in question, some of the included modules may be missing, and others may themselves take on a complicated structure. However, the supplied model suffices to illustrate many of the issues in sensing and estimation. In the remainder of this section, we discuss an example from mobile localization to illustrate this model.

The initial problem in sensory processing is data preprocessing and feature extraction. The role of preprocessing is to reduce noise from the transducer, to remove any systematic errors, and to enhance relevant aspects of the data. In some cases, sensory information might also have to be temporally or spatially aligned for subsequent integration. There are innumerable ways

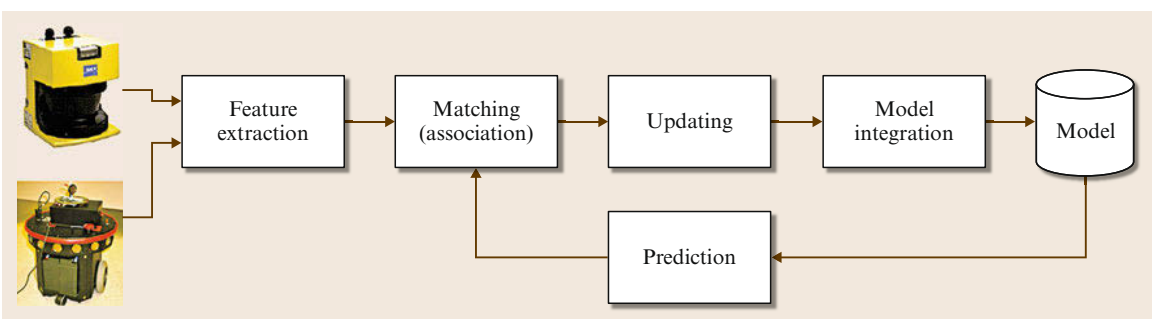


Fig. 5.1 Example of a perception process as discussed in this chapter

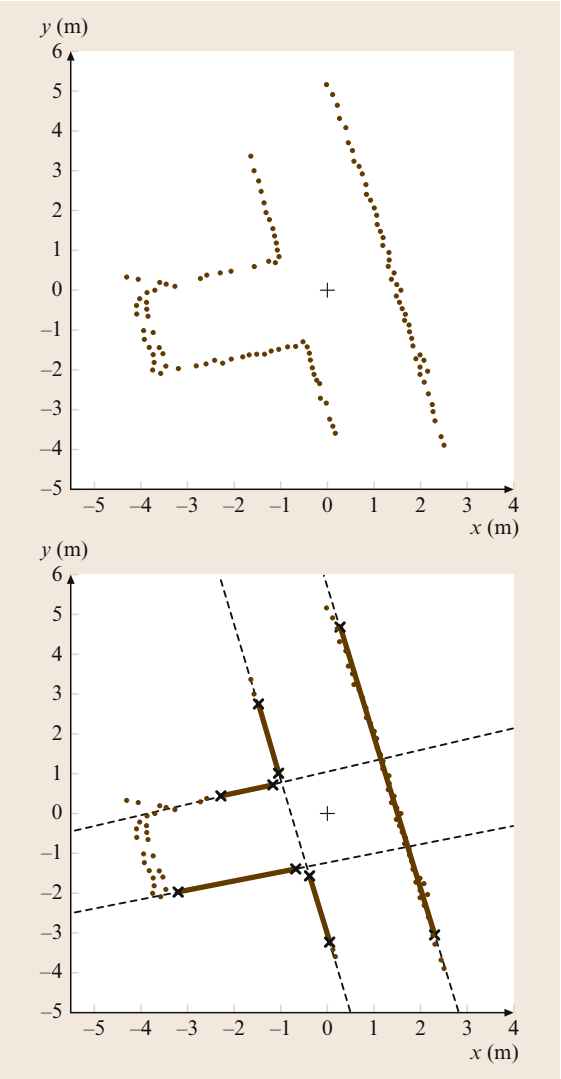


Fig. 5.2 An example of feature extraction from a laser scan (after [5.2])

that data can be preprocessed to enhance or extract features that are used in the integration. One common approach is model fitting, as illustrated for a laser scanner in Fig. 5.2. Once sensor information is available, it is often necessary to match the data with an existing model (Fig. 5.3). This model may be based on a priori known structure (e.g., a computer-aided design (CAD) model of the environment), or may have been built up from previously acquired data. Data association methods are commonly employed to estimate the relationship between sensor data and the model of the environment. In our mobile robot localization example, the extracted line features are matched against a polygonal world model. This matching process can be performed in several different ways, but in general it is an optimization that maximizes the alignment of features to the model.

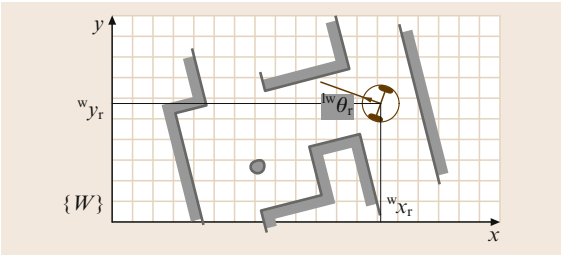


Fig. 5.3 An example environmental model for mobile robot localization (after [5.2])

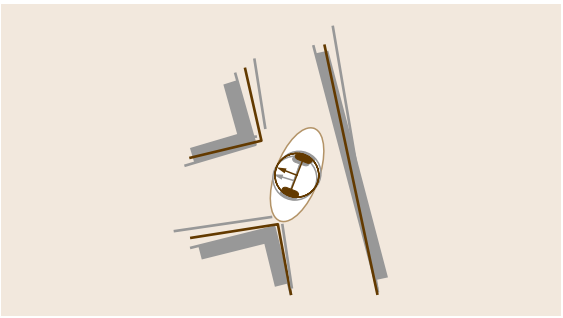


Fig. 5.4 Estimation of position and orientation for the example mobile robot (after [5.2])

Once sensory data has been matched against the world model it is possible to update the model with new information contained in the sensor data. In the example, the orientation and position of the robot relative to the world model can be updated (Fig. 5.4) from the matched line segments.

Finally, it may be possible to develop a dynamical system model of the underlying state being estimated. Using such a system model, it is possible to predict how the world changes over time until new sensory data is acquired. This can be used within a feed-forward prediction process, which in turn simplifies data association for new sensory readings, as shown in Fig. 5.1.

With this as a prologue, we now turn to discuss each step of the perception process in greater detail.

Avidbots Ex1016
Page 139 of 2154

5.3 Sensors

There are a variety of ways to classify sensors depending on what they measure, and how they measure it. As noted previously, proprioceptive sensors are used to measure the internal state of a robot, which might include position of different degrees of freedom, temperature, voltage on key components, motor current, force applied to an effector, and so forth. Exteroceptive sensors, on the other hand, generate information about the external environment in terms of distance to an object, interaction forces, tissue density, and so forth.

Sensors may also be differentiated based on whether they are passive or active. In general, an active sensor is one that emits energy into the environment, and measures properties of the environment based on the response. A passive sensor is one that is not active. Active sensors are generally more robust than passive sensors since they exert some control over the measured signal. For example, a passive stereo camera system must rely on the appearance of viewed surfaces when performing feature matching for triangulation (Chap. 31), whereas structured light systems project a pattern onto the scene and are thus less sensitive to scene characteristics. Even so, absorbtion, scattering or interference of the emitted signal can affect the performance of active sensors.

Proprioceptive sensors are typically passive and usually measure physical properties of the robot such as joint position, velocity, or acceleration, motor torque, and so forth. Exteroceptive sensors, on the other hand, can be further divided into contact and noncontact sensing. The contact sensors are typically the same modalities as used for proprioception, while noncontact sensor sensors involve most of the modalities that can be used for estimation of physical properties at a distance including intensity, range, direction, size, and so forth.

A classification of typical sensors according to method and typical application is shown in Table 5.1. More detail on methods of sensing, characterization of sensors, and general applications can for example be found in the Handbook of Modern Sensors [5.3] and in Part C of this handbook.

Estimation of rotational motion is fundamental to control of robot manipulators and also for estimation of ego-motion for mobile systems. The most common sensor for measurement of rotation is the quadrature encoder. It is composed of a transparent disc, with two periodic patterns that are out of phase, as shown in Fig. 5.5. Through the use of counters it is possible

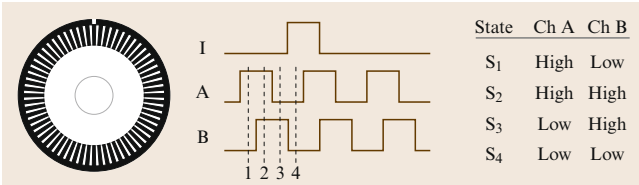


Fig. 5.5 Sketch of the quadrature encoder disc, and output from photodetectors placed over each of the two pattern. The corresponding state changes are shown on the *right*

Table 5.1 Classification of sensors frequently used in robotics according to sensing objective (proprioception (PC)/exteroception (EC)) and method (active/passive)

Classification	Sensor type	Sens	A/P
Tactile sensors	Switches/bumpers	EC	P
	Optical barriers	EC	A
	Proximity	EC	P/A
Haptic sensors	Contact arrays	EC	P
	Force/torque	PC/EC	P
	Resistive	EC	P
Motor/axis sensors	Brush encoders	PC	P
	Potentiometers	PC	P
	Resolvers	PC	A
	Optical encoders	PC	A
	Magnetic encoders	PC	A
	Inductive encoders	PC	A
	Capacity encoders	EC	A
Heading sensors	Compass	EC	P
	Gyroscopes	PC	P
	Inclinometers	EC	A/P
Beacon based (postion wrt an inertial frame)	GPS	EC	A
	Active optical	EC	A
	Radio frequency (RF) beacons	EC	A
	Ultrasound beacon	EC	A
	Reflective beacons	EC	A
Ranging	Capacitive sensor	EC	P
	Magnetic sensors	EC	P/A
	Camera	EC	P/A
	Sonar	EC	A
	Laser range	EC	A
	Structured light	EC	A
Speed/motion	Doppler radar	EC	A
	Doppler sound	EC	A
	Camera	EC	P
	Accelerometer	EC	P
Identification	Camera	EC	P
	Radio frequency identification RFID	EC	A
	Laser ranging	EC	A
	Radar	EC	A
	Ultrasound	EC	A
	Sound	EC	P

Avidbots Ex1016
Page 140 of 2154

to directly compute the motion and its direction (the phasing between sensors A and B in Fig. 5.5). In addition the disc is frequently fitted with a single dot on the outer rim for indexing (specification of a zero index). The density of the pattern determines the resolution of the measurements. When fitting the sensor to a motor before a reduction gear it is easy to achieve accuracies beyond $1/1000°$.

For the estimation of force and torque at an end-effector it is possible to use piezoelectric elements. These elements generate a voltage that is proportional to the introduced deformation. Through careful placement it is possible to measure both force and torque. The sensors are used in robotic manipulation as part of assembly systems, deburring, etc. and also in medical applications for the estimation of stress and contact. Force/torque sensors are widely available in a range of sizes and dynamic ranges, including new flexible arrays that can be mounted on a variety of end-effectors (Fig. 5.6). These arrays are more commonly referred to as *tactile sensors* as they begin to simulate the human tactile sense. See Chap. 19 and [5.4, 5] for recent reviews of the state of the art in tactile sensing. Potential problems with force sensors are a dead band on initial contact, and noisy data from the basic sensing elements, which calls for signal processing to clean up the data.

Ego-motion estimation is an important part of almost all robotic systems. To this end it is possible to use inertial measurement units (IMU). An IMU typically includes both accelerometers and gyros. Accelerometers are sensitive to all types of acceleration, which implies that both translation motion and rotation (centripetal forces) are measured in combination. Joint IMU units allow the estimation of rotation and translation, and allow for double integration to estimation the velocity, orientation, and position of a system, as for example reported in [5.6]. One of the problems associated with the use of an IMU is the need for double integration. Small biases and noise can result in significant divergence in the final estimate, which calls for use of detailed models and careful calibration and identification of sensor characteristics. An example of data from a cross-bow DMU-6x unit for a car driving on an unpaved road is shown in Fig. 5.7.

Much early work on mobile robotics, underwater robots, and some medical robotics relies on ultrasonic ranging. The general class of sensors are often termed sound navigation and ranging (sonars). The general principle is that the system emits a sound pulse and awaits the return of echoes that have bounced off objects in the environment. Knowing the transmission speed in the medium and the time of flight it is possible to compute the distance. The method was widely used in early robotics due to the availability of low-cost sensors with adequate performance. In underwater robotics this is still a primary sensor. Sonar is discussed in detail in Chap. 30.

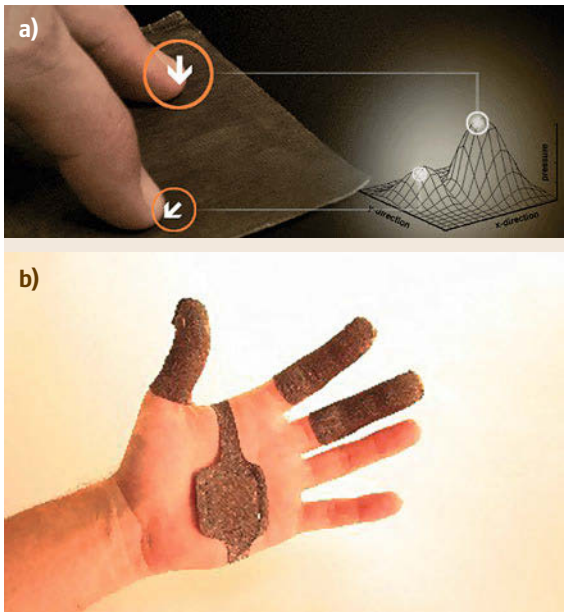


Fig. 5.6 **(a)** TactArray, a flexible capacitive array tactile sensor from Pressure Profile Systems, Inc., is appropriate for sensing contact locations and areas under sliding conditions. **(b)** Conformable TactArray sensors can fit on a human or robotic hand (courtesy Pressure Profile Systems, Inc.)

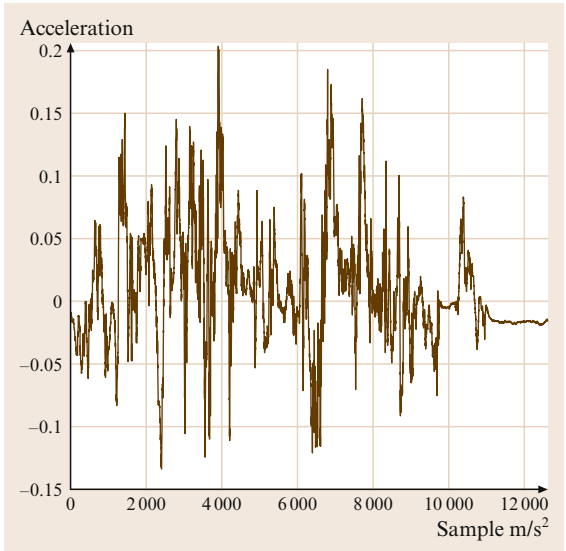


Fig. 5.7 Example data from an IMU unit for driving on an unpaved road



Fig. 5.8 Example laser ranging sensor (SICK LMS291) that is widely used in mobile robotics

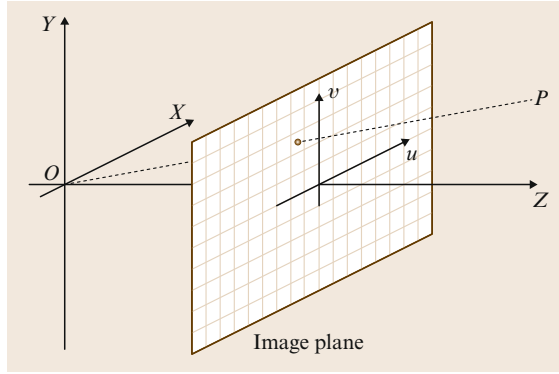


Fig. 5.9 The pinhole camera model



Fig. 5.10 Catadioptric image and the same image mapped to a cylindrical surface

Recent progress on environmental modeling and navigation has, in many respects, been due to the emergence of low-cost high-fidelity laser scanning systems. The SICK series of laser scanners are time-of-flight scanners. The scanner sends out a pulse of light and measures the time to return. The standard scanner enables estimation of distances up to 80 m at centimeter or millimeter accuracy. The scanner measures distances in a plane with an angular resolution of 0.5−1°. The field of view is 180° resulting in 181−361 range measurements. The sensor data are contaminated by uniformly distributed noise, which must be considered in the detection of features or integration of data into a raw sensor map (Fig. 5.8).

Imaging sensors are a rich source of information for sensing and estimation. Imaging sensors come in a wide variety of configurations, varying according to imaging geometry, image resolution, sensor technology and the range of sensed spectral bands. Most readers are no doubt familiar with the traditional three-CCD, perspective color camera. In this case, there are three charge-coupled detector (CCD) arrays, each receiving a portion of the visible spectrum corresponding roughly to the human perception of red, green, and blue colors. A common and less expensive alternative is a so-called single-chip CCD camera. In this case, a special spatial array of color filters, usually referred to as a Bayer filter after its inventor Bryce Bayer, is employed. The resulting spatial array is subsequently processed (a process referred to as demosaicing) to provide color information for each pixel.

In the United States, image sensors traditionally contained 480 rows of 640 pixels according to the National Television System Committee (NTSC) standard created for analog transmission of television signals. The corresponding European standard, PAL, has 576 lines of 768 pixels. More recently, the advent of digital interfaces such as IEEE 1394 and USB 2.0 have allowed camera systems to be developed with significantly improved resolution ranging into the millions of pixels. At the same time, cost-effective infrared (IR) and ultraviolet (UV) cameras have become available, allowing the development of advanced multispectral image interpretation systems.

A traditional imaging sensor contains an optical system that focuses light on a planar imaging array. In most cases, this system can be modeled using the classical pinhole camera model (shown in Fig. 5.9). Given

Avidbots Ex1016
Page 142 of 2154

a point $(x, y, z)^{\mathrm{T}}$ in Euclidean space, the corresponding camera pixel coordinates $(u, v)^{\mathrm{T}}$ are given by

$$(u - u_{\mathrm{c}}) = \frac{f}{s_x}\frac{x}{z}\,,$$
$$(v - v_{\mathrm{c}}) = \frac{f}{s_y}\frac{y}{z}\,, \qquad (5.1)$$

where f is the focal length of the lens system, u_{c} and v_{c} are the pixel coordinates of the center of projection, and s_x and s_y are the size of a single pixel on the imaging array. In practice, these models are also augmented with low-order models of image distortion. The values of these parameters for a given camera system can be determined experimentally using a variety of methods [5.7].

By combining a traditional perspective camera with a mirror, creating a so-called catadioptric system, it is possible to create imaging geometries that map fields of view as large as a hemisphere into a single image. Such systems are useful, for example, for surveillance, and their geometric properties provide for stable position referencing for mobile navigation [5.8]. An example image is shown in Fig. 5.10 together with the corresponding image when mapped to a cylindrical surface.

Active ranging cameras, which combine images from an optical camera with a dense range map, are now also widely available and quite cost effective. These systems perform triangulation between a light projector that throws a pattern with known structure into the scene, and a camera that views the scene and detects the pattern. By matching or correlating the pattern elements between the projector and the camera, and using the known relationship between them, it is possible to recover depth using standard methods. The camera/projector pair for depth recovery typically operate in the infrared to avoid the pattern being visible to the human eye. Thus, these systems typically include a third visible light camera to acquire a corresponding color image. Hence the name RGB-D (red–green–blue–depth) cameras. Fig. 5.11 shows two recent low-cost RGB-D products. In Fig. 5.12 is show the fused output for a tabletop scene.

The discussion above has touched on the most commonly employed robotic sensing devices. Many special-purpose sensors are employed for specific applications. In medicine (Chap. 63), ultrasound, X-ray, computed tomography, and magnetic resonance imaging are commonly employed. Underground mapping makes use of ground-penetrating radar [5.9]. Underwater robotics makes use of many variations on acoustical sensors. Further discussions of these more task-specific sensing modalities can be found in the application chapters in Part C of this handbook.



Fig. 5.11 Picture of proxim and kinect RGB-D sensors

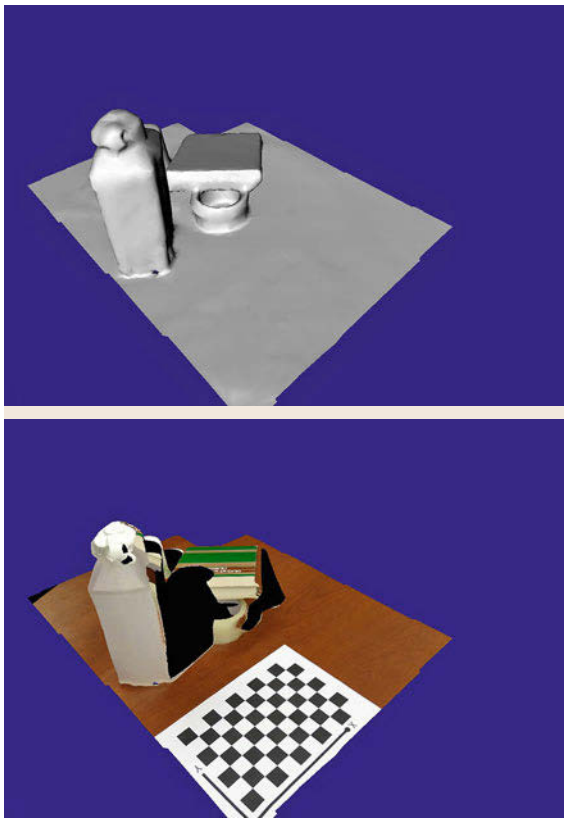


Fig. 5.12 Example of fused range images from RGB-D camera and the same scene as an intensity textured mesh

5.4 Estimation Processes

As discussed in the introduction, there are many different techniques for combining information from sensors. The appropriate set of techniques depends, to a great degree, on what is known a priori about the environment, what information is necessary for the task at hand, and what models for the sensing system are appropriate. Common methodologies include simple voting-based methods, parametric and nonparametric statistical estimation techniques, fuzzy logic-based systems, and Dempster–Shafer theory.

To illustrate this point, consider the robot localization problem introduced in Sect. 5.2. At the outset, if nothing is known about the environment, the robot may acquire a laser scan and try to produce an initial model of the environment using line segments. Since nothing is known a priori, the system must estimate:

1. The number of line segments
2. The data association between line segments and observed data values
3. The parameters of the line segments themselves.

This is a challenging problem that can be attacked by simple voting techniques such as the Hough transform [5.10] or random sample consensus (RANSAC) [5.11] or more sophisticated unsupervised clustering methods such as *k*-means [5.12], expectation maximization (EM) [5.13], or generalized principal component analysis (GPCA) [5.14]. In many cases, this is a computationally intensive, iterative process.

Conversely, if a prior CAD model for the environment is known, then the problem is to produce a small set of parameters (translation and rotation) of the model to match the data. This problem can be solved, using feature matching by aligning observed points to the model with iterative closest-point algorithms (ICP) [5.15] or other efficient combinatorial matching algorithms such as Monte Carlo methods [5.16]. The best method to apply again depends to a great degree on the structure of the environment and what is known a priori.

Once an initial registration is known, new data can take advantage of the fact that strong prior knowledge is available. In particular, as the robot moves, the sensor data should change in a predictable fashion. Thus is it possible to make use of predictor–corrector methods such as the Kalman filter [5.17, 18] or sequential importance sampling [5.19], provided appropriate statistical characterizations of the sensing system are available. The data association problem, if present, can be addressed using a variety of general techniques such as EM [5.12] or more specialized modifications to the previously cited predictor–corrector methods [5.20].

It is often the case that sensor data is corrupted by occasional nonsensical values. For example, the laser range finder in our example may occasionally return a spurious range value due to a reflection. Many commonly used estimation techniques are not robust to such so-called data outliers. Techniques from robust statistics [5.21] can be used to improve the performance of sensing and estimation systems in such cases.

Finally, we may want to consider what information is actually important for the task at hand. Most of the techniques above presume that the goal is to produce an accurate estimate of a set of continuous parameters closely related to the underlying data. However, in some tasks, the parameter values themselves may not be what is of interest. For example, suppose that the goal of our robot is to drive through a doorway. Although this clearly depends on an ability to estimate the width of the door (a continuous parameter), the decision is ultimately binary. This problem can be codified as a *decision problem*. Decision problems can be modeled using concepts from *decision theory* [5.22] including zero–one loss functions, likelihood ratios, or probability ratios. For example, in the case of fitting through a door, for a low-priority task there may be a low cost associated with not attempting to move through this particular door (necessitating replanning to find an alternative route) relative to attempting to navigate through an opening that is too small (risking damage to the robot or the door, or both). Conversely, if the task is urgent, more risky behavior may be warranted.

For any given task (or decision), the amount of information necessary to reach the decision may vary, for example, if the doorway is quite wide, it may require relatively little information to safely navigate through it. Conversely, a tight fit may require close inspection before a decision can be reached. The problem of determining the type and/or amount of information necessary to reach a decision is referred to variously as the *sequential sampling problem* [5.22], the sensor control problem, or the sensor planning problem [5.23–25].

5.4.1 Point Estimation

In our robot localization example we saw several cases where the key problem was to estimate an unknown quantity that can be represented as a point in a vector space. Examples include the location of a two-dimensional (2-D) or three-dimensional (3-D) point or the location of a robot. We also saw examples where the problem was to locate the pose (position and orien-

Avidbots Ex1016
Page 144 of 2154

tation) of the robot, or parameters of a line segment. The latter differ in that the underlying parameter space is not a vector space. This introduces some additional unique problems. We refer the reader to [5.26, 27] for further discussion, and in the remainder of this chapter restrict our attention to *point estimation* problems on vector spaces. In our discussion, we assume the reader is familiar with multivariate Gaussian distributions as described in [5.28] and basic linear algebra [5.29].

In the remainder of this section, we consider the following general problem.

Given: an observation model

$$y = f(x, \eta) \,. \tag{5.2}$$

Estimate: $x \in \mathrm{Re}(n)$ from observations $y \in \mathrm{Re}(m)$ where η is an unknown disturbance taking values in $\mathrm{Re}(k)$ and f is a known mapping from $\mathrm{Re}(k+n)$ to $\mathrm{Re}(m)$.

We divide our discussion into two topical areas:

- Methods for performing estimation on batch and sequential data when f is linear,
- Methods for performing estimation on sequential data when f is nonlinear.

Estimation Techniques for Batch and Sequential Data with Linear Models

In this section, we discuss linear and linearized estimation techniques for sequential data, including the Kalman filter and extensions thereof. Our goal is to provide an overview of techniques available. The reader may also wish to consult more in-depth references such as [5.18, 30, 31] and (Chap. 35) for additional information.

We first consider the case when f in (5.2) is linear in its arguments. In this case, we can write

$$y = \mathbf{F}x + \mathbf{B}\eta \,, \tag{5.3}$$

where $\mathbf{F} \in \mathrm{Re}(m \times n)$ defines the (linear) relationship between the unknown x and the observation y and $\mathbf{B} \in \mathrm{Re}(m \times m)$. For the moment, we will drop $\mathbf{B}$ and assume that η represents the complete disturbance model of the system.

The *least-squares* method of estimating x from y proceeds by solving the optimization problem

$$\min_x \|\mathbf{F}x - y\|^2 \,. \tag{5.4}$$

This optimization has a unique solution $\hat{x}$ if and only if the matrix $\mathbf{F}$ has full column rank. In this case, the solution can be computed by solving the following linear system

$$\mathbf{F}^{\mathrm{T}}\mathbf{F}\hat{x} = \mathbf{F}^{\mathrm{T}}y \,. \tag{5.5}$$

In some cases, there may be reason to believe that some observed elements are more reliable than others, and hence should contribute more to the final estimate. This information can be incorporated by modifying (5.4) to include a diagonal positive-definite weighting matrix $\mathbf{W}$ as

$$\min_x (\mathbf{F}x - y)^{\mathrm{T}}\mathbf{W}(\mathbf{F}x - y) \,. \tag{5.6}$$

The solution is then given by solving

$$(\mathbf{F}^{\mathrm{T}}\mathbf{W}\mathbf{F})\hat{x} = \mathbf{F}^{\mathrm{T}}\mathbf{W}y \,. \tag{5.7}$$

Although (5.3) included a disturbance component (in the form of η), the parameter estimates computed in (5.5) or (5.7) made no explicit use of this quantity. However, we can often model the noise characteristics of the underlying sensor using a statistical model and recast our original estimation problem to incorporate this information. One common method is to compute the *maximum-likelihood estimate* (MLE), which is a value $\hat{x}$ such that

$$p(y|\hat{x}) = \max_x p(y|x) \,. \tag{5.8}$$

For the linear additive model of (5.3), the likelihood function can be expressed in a particularly simple form. Suppose that η is described by a fixed, known probability density function D. The likelihood function is then given by

$$p(y|x) = D(y - \mathbf{F}x) \,. \tag{5.9}$$

The MLE can be related to the previous least-squares method as follows. Suppose that $\eta \sim N(0, \mathbf{\Lambda})$, where N denotes a multivariate Gaussian density function with (mean) 0 and covariance $\mathbf{\Lambda}$. Upon observing that the maximizing the value of the likelihood function is equivalent to minimizing the negative log of the likelihood function, a short series of calculations shows that the optimal maximum-likelihood estimate is computed by weighted least squares with $\mathbf{W} = \mathbf{\Lambda}^{-1}$.

Finally, there is often a reason to include the idea that some parameters are more likely a priori to occur as others. For example, when observing a car driving on an expressway, a velocity of 60 mph is much more likely than either 20 or 300 mph. This information can be captured in *prior statistics* on the unknown value x.

Avidbots Ex1016
Page 145 of 2154

Given a prior probably density on x, $p(x)$, *Bayes theorem* states that

$$p(x|y) = \frac{p(y|x)p(x)}{p(y)} = \frac{p(y|x)p(x)}{\int p(y|x)p(x)\,\mathrm{d}x} . \tag{5.10}$$

The *maximum a posteriori probability* (MAP) estimate is the value $\hat{x}$ such that

$$p(\hat{x}|y) = \max_x p(x|y) . \tag{5.11}$$

In general, the solution to this optimization problem can be quite complex. Rather than pursue this course further, we consider another alternative. Namely, provided the second moments of $p(x|y)$ exist, it is possible to produce a *least-squares* estimate, in a statistical sense, by solving the following optimization problem over an unknown function δ

$$\min_{\delta} E\,\|\delta(y) - x\|^2 . \tag{5.12}$$

That is, the best function δ is one that produces an estimate of x from y with *minimum mean-square error* (MMSE). Thus, the estimator δ is often referred to as an MMSE estimator.

It can be shown that, in the general case, the *optimal* decision rule δ^* is the conditional mean [5.22]

$$\delta^*(y) = E\,[x \mid y] . \tag{5.13}$$

Unfortunately, this expression, as with the MAP estimate defined above, can be extremely difficult to compute for the general case. Later we consider methods for computing approximations to (5.13). For now, we again consider our previous linear observation model (5.3) (without B). Additionally, we suppose that x and η are *independent* random variables with finite second moments, and both are *zero-mean* random variables. Note that the latter is not really a restriction since it can be accomplished by simply defining a new variable $x' = x - E[x]$. Finally, we will consider only *linear* functions δ, that is, we can write $\hat{x} = \delta(y) = \mathbf{K}y$.

With this, (5.12) can be expanded as

$$\begin{aligned} E\,\|\delta(y) - x\|^2 &= E\,\|\mathbf{K}y - x\|^2 \\ &= E\,\|\mathbf{K}(\mathbf{F}x + \eta) - x\|^2 \\ &= E\,\|(\mathbf{K}\mathbf{F} - I)x\|^2 + E\,\|\mathbf{K}\eta\|^2 \\ &= \mathrm{tr}\left[(\mathbf{K}\mathbf{F} - I)\mathbf{\Lambda}(\mathbf{K}\mathbf{F} - I)^{\mathrm{T}} + \mathbf{K}\mathbf{\Sigma}\mathbf{K}^{\mathrm{T}}\right] . \end{aligned} \tag{5.14}$$

Here, the independence of x and η and the fact that they are both zero mean has eliminated several terms. The final step makes use of the fact that $\|x\|^2 = \mathrm{tr}(xx^{\mathrm{T}})$.

Taking derivatives with respect to $\mathbf{K}$ and setting them equal to zero yields the solution

$$\mathbf{K} = \mathbf{\Lambda}\mathbf{F}^{\mathrm{T}}(\mathbf{F}\mathbf{\Lambda}\mathbf{F}^{\mathrm{T}} + \mathbf{\Sigma})^{-1} . \tag{5.15}$$

Thus, in this case the optimal estimate is given by a linear function of the observation, where the linear term depends only on the variance of the underlying random variables and the linear term defining the observation system.

If x is not zero-mean, but has mean μ, it is not hard to show that the optimal estimate is

$$\hat{x} = \mathbf{K}y + (\mathbf{I} - \mathbf{K}\mathbf{F})\mu , \tag{5.16}$$

and that the variance of the estimate Λ^+ is

$$\mathbf{\Lambda}^+ = (\mathbf{I} - \mathbf{K}\mathbf{F})\,\mathbf{\Lambda} . \tag{5.17}$$

The interested reader may wish to work this out for a few simple cases, for example, if $\mathbf{\Lambda} = \mathbf{\Sigma}$ and $\mathbf{F} = \mathbf{I}$, $\mathbf{K} = 1/2\mathbf{I}$ and thus $\hat{x} = y + \mu$ – a simple average – with variance $\mathbf{\Lambda}^+ = 1/2\mathbf{\Lambda}$.

When both the observation noise and prior statistics are Gaussian distributions, then it can be shown that the solution we have derived is also the MAP estimate for the unknown x [5.22].

The Kalman Filter

With this as background, we are now in a position to define the discrete-time Kalman–Bucy filter [5.32] for linear systems. Consider the following time-series model

$$x_{t+1} = \mathbf{G}x_t + w_t , \tag{5.18}$$

$$y_t = \mathbf{F}x_t + \eta_t , \tag{5.19}$$

where $\mathbf{G}$ is an $n \times n$ matrix describing the system time evolution and x_0 is distributed according to a Gaussian distribution with mean $\hat{x}_0$ and variance $\mathbf{\Lambda}_0$. In addition w_t and η_t are zero-mean Gaussian independent random variables for all t, w_t is independent of $w_{t'}$ for all $t \neq t'$, and likewise η_t is independent of $\eta_{t'}$ for all $t \neq t'$. Finally, η_t has variance $\mathbf{\Sigma}_t$ and w_t has variance $\mathbf{\Omega}_t$.

Given an observation y_1 it is possible, using the derivation of the previous section, to compute an updated estimate $\hat{x}_1$ with variance $\mathbf{\Lambda}_1$. Note, that the solution is a linear combination of two Gaussian random variables: the observation value y_1 and the prior estimate $\hat{x}_0$. As any linear combination of Gaussian random variables is also a Gaussian random variable, it follows that the updated estimate is also Gaussian.

Avidbots Ex1016
Page 146 of 2154

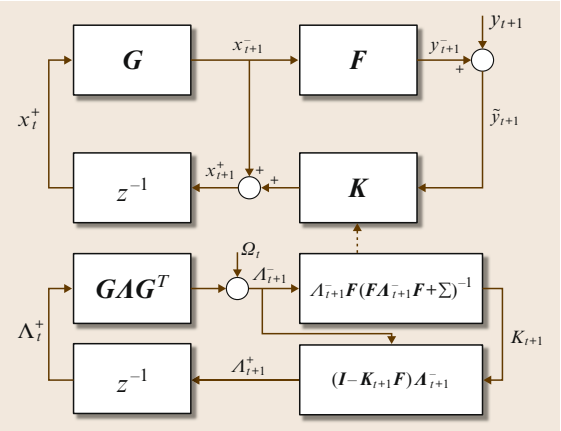


Fig. 5.13 A summary of the Kalman filter

Now, we add one additional step: projection through the dynamics model. To describe this, superscripts minus and plus will denote before and after the estimation step, respectively. Thus, given an estimate $\hat{x}_t^+$ with variance $\mathbf{\Lambda}_t^+$, the projection ahead one time step produces

$$\hat{x}_{t+1}^- = \mathbf{G}x_t^+ \,, \tag{5.20}$$

$$\mathbf{\Lambda}_{t+1}^- = \mathbf{G}\mathbf{\Lambda}_t^+\mathbf{G}^{\mathrm{T}} + \Omega_t \,. \tag{5.21}$$

At this point, a new observation y_{t+1} is acquired and the cycle repeats. The summarization of the complete Kalman filtering algorithm for linear systems is shown in Fig. 5.13.

It is possible to show that the Kalman filter is the *optimal* filter, under the stated assumptions, in the mean-square sense. It is also the optimal *linear* filter when either or both Gaussian assumptions do not hold.

Nonlinear Estimation Techniques for Sequential Data

The results of the previous subsection presume a linear form for the relationship between the observation and system state, additive noise, and a linear relationship describing the state evolution. Furthermore, the stated results are globally optimal for systems with Gaussian observation and driving noise, but are only the best *linear* estimator if the noise sources are non-Gaussian.

As noted at the outset, the more general nonlinear (discrete-time) system description is

$$\begin{aligned} x_{t+1} &= g_t(x_t) + w_t \,, \\ y_t &= f_t(x_t) + \eta_t \,, \end{aligned} \tag{5.22}$$

where, for the moment, the noise model continues to be additive.

Although this model contains nonlinear elements, it is still possible to apply a variant of the Kalman filter, the extended Kalman filter (EKF) by making use of the Taylor-series expansion of the nonlinear elements about the current estimates. Let $\mathbf{J}_f$ (resp. $\mathbf{J}_g$) denote the Jacobian matrix of the function f (resp. g). Supposing that an estimate at time step $t-1$ exists, the first-order expansion of (5.22) about this point yields

$$x_{t+1} = g_t(\hat{x}_{t-1}) + \mathbf{J}_g(\hat{x}_{t-1})(x_t - \hat{x}_{t+1}) + w_t \,, \tag{5.23}$$

$$y_t = f_t(\hat{x}_{-1}) + \mathbf{J}_f(\hat{x}_{t-1})(x_t - \hat{x}_{t-1}) + \eta_t \,. \tag{5.24}$$

Rearranging yields a linear form appropriate for the previously defined Kalman filter

$$\tilde{x}_{t+1} = x_{t+1} - g_t(\hat{x}_{t-1}) + \mathbf{J}_{g_t}\hat{x}_{t-1} = \mathbf{J}_{g_t}x_t + w_t \,, \tag{5.25}$$

$$\tilde{y}_t = y_t - f_t(\hat{x}_{t-1}) + \mathbf{J}_{f_t}\hat{x}_{t-1} = \mathbf{J}_{f_t}x_t + \eta_t \,. \tag{5.26}$$

In this form, $\tilde{x}$ and $\tilde{y}$ are new *synthetic* state and observation variables, $\mathbf{J}_g(\hat{x}_{t-1})$ plays the role of G, and $\mathbf{J}_f(\hat{x}_{t-1})$ plays the role of F.

It is worth noting that the EKF iterations are essentially a form of weighted Newton iterations (i. e., an iterative nonlinear estimation method). As a result, it is often useful to iterate more than once *on the same observation* while holding the variance terms fixed. This allows the estimator to converge to a solution in the presence of large disturbances or significant nonlinearities. Only after convergence are the variance terms updated. This version of the Kalman filter is referred to as the *iterated extended Kalman filter* (IEKF).

5.4.2 Other Approaches to Estimation

In the previous section, we reviewed a common and widely used estimation method. However, there are several alternative methodologies for solving parameter estimation problems. Here we briefly introduce two: sequential importance sampling and graphical models.

Sequential Importance Sampling

Much of the discussion heretofore has centered around the notion of approximating everything known about the system state using an estimated mean and covariance. An alternative presents itself by simply going back to Bayes theorem which states, in general, that

$$p(x_n|y_1, y_2 \ldots y_n) = \frac{p(y_1, y_2 \ldots y_n|x_n)p(x_n)}{p(y_1, y_2 \ldots y_n)} \,. \tag{5.27}$$

Assuming that y_n is independent of all prior observations and states given x_n, and that x_n is independent of x_{n-k} for $k > 1$ given x_{n-1}, this expression simplifies

Avidbots Ex1016
Page 147 of 2154

to

$$p(x_n|x_{n-1}, y_n) = \frac{p(y_n|x_n)p(x_n \mid x_{n-1})}{p(y_n \mid x_{n-1})} . \quad (5.28)$$

Recall that the optimal mean-square estimate is given by the conditional mean which, in this case, is

$$\delta^*(y_n) = E[x_n \mid y_n] . \quad (5.29)$$

In fact, we essentially showed that the Kalman filter is a special case of this result for linear systems corrupted by Gaussian noise.

The difficulty in implementing this procedure in the general case ultimately comes down to the problem of representing and computing with the distributions that arise in the nonlinear, non-Gaussian case. However, suppose that the heretofore continuous variable x_n only took on a discrete set of values. In this case, computing Bayes theorem and other associated statistical quantities reduces to a straightforward set of computations on this discrete set of variables. This can be simply done for any distribution and any set of transformations.

Sequential important sampling (also known as particle filtering, condensation, and a variety of other names) is a way of applying this approach to continuous variables in a statistically sound manner. In order to perform sequential importance sampling, it is assumed that:

1. It is possible to *sample from* the likelihood function $P(y_n \mid x_n)$, and
2. It is possible to *sample from* the dynamical model $P(x_n \mid x_{n-1})$.

Note the emphasis on sampling – there is no need to explicitly exhibit an analytical form of the likelihood function or of the dynamical model.

Given this, sequential important sampling, in its simplest form, can be written as follows:

1. Let $\pi_{n-1} = \{\langle x^k_{n-1}, w^k_{n-1}\rangle, k = 1, 2, \ldots N\}$ represent a set of sample points x^k_{n-1} together with a set of weights w^k_{n-1} with $\sum w^k_{n-1} = 1$.
2. Compute a new set of N samples $\pi^-_{n-1} = \{\langle x^k_n, 1/N\rangle, k = 1, 2, \ldots N\}$ as:
   a) Choose a sample point x^{k-1}_{n-1} with probability proportional to its weight w^{k-1};
   b) Sample from $P(x_n \mid x^k_{n-1})$ given x^k_n with weight $1/N$;
3. Compute $\pi_n = \{\langle x^k_n, P(y_n \mid x^k_n)\rangle, k = 1, 2, \ldots N\}$.

It is easy to see that this set of steps is now in the form of a recursive filter. Furthermore, at any time any statistic of the associated distribution can be approximated from the set of samples and associated weights.

Sampling-based filters of this form have found wide applicability in a variety of challenging areas where linear estimation techniques do not suffice. These techniques have been particularly successful, for problems with low state dimension (typically $n \leq 3$) and well-constrained dynamics. For higher-dimensional problems or systems exhibiting high dynamic variability, the number of particles necessary to obtain good approximations can become prohibitively large. However, even in these cases, sampling-based systems can sometimes be engineered to produce acceptably good results.

Graphical Models

Graphical models are a class of models that represent dependence and independence relationships among a set of variables. Common examples of graphical models include Bayes nets, influence diagrams, and neural nets. Graphical models are quite general – indeed much of this chapter could have been written by first defining graphical models, and exploring specializations that lead to the Kalman Filter, for example. Here, for reasons of space, we focus on Bayes nets as a specific example of graphical models.

A Bayesian network is a directed acyclic graph consisting of nodes representing random variables, and directed arcs representing probabilistic relationships between pairs of random variables. Let parents(X) denote the set of nodes which have arcs terminating at X, and let $X_1, X_2, \ldots, X_N$ be the N random variables in the graph. Then we can write

$$P(X_1, X_2, \ldots, X_N) = \prod_{i=1}^{N} P(X_i \mid \text{parents}(X_i)) . \quad (5.30)$$

For example, a Bayesian network representing a mobile robot performing localization is shown in Fig. 5.14. This graphical model encodes the sequential form of the problem and is thus an example of a so-called *recurrent*

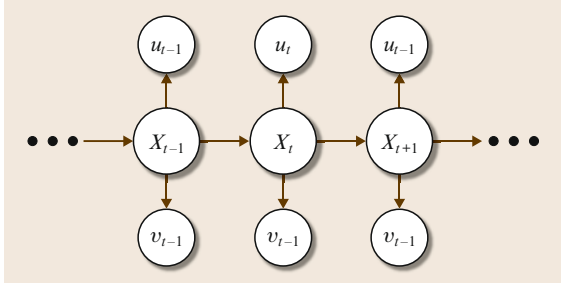


Fig. 5.14 An example of robot localization expressed as a graphical model

Avidbots Ex1016
Page 148 of 2154

network. More discussion of such models can be found in [5.33].

The structure of a Bayesian network encodes various independence relationships among variables. By exploiting these independence relationships, it is possible to design efficient inference algorithms. In particular, graphs which are acyclic even in their undirected form (referred to as *polytrees*) admit linear time inference algorithms. More general graphs can be solved using various types of iterative methods. In particular, if the distributions in the network are of a continuous type, variations on sequential importance sampling can be used to solve problems in an approximate sense [5.34].

Conditional Random Fields

In many cases of interest, including many of the examples in this chapter, the end goal is to infer or predict a value or label from observed data. We might then frame the problem by exploring the joint distribution $P(X, Y)$ where X represents some data that is *observed* and Y is what we would like to infer. Recall that

$$P(X, Y) = P(Y|X)P(X) \, .$$

If we take $X = x$ for some observed values x, then we see that $P(X)$ becomes constant, and inferring a value for Y depends only on $P(Y|X)$. If we were to apply a Bayes Net to this problem, the model would represent the complete joint probability distribution on X and Y, what is referred to as a *generative model*. But, if we know X is always observed, then much of this structure is irrelevant to our problem – we don't care about the probabilistic structure of X. This observation has given rise to a specialization of graphical models referred to as *Conditional Random Fields*, or CRFs for short.

The immediate value of CRFs is their economy and expressivity compared to graphical models. This has immediate positive implications for the complexity of both learning and inference. Traditionally CRF models are learned using maximum likelihood-based methods using gradient descent or other unconstrained optimization techniques. However, recent methods like Cutting Planes [5.35] and Block Coordinate Frank Wolfe [5.36] pose it as a constrained optimization problem in the form of a Structural Support Vector Machine. These techniques tend to be more computationally efficient and are often more accurate.

CRFs have proven to be very general, and are now extremely widely used for image processing, natural language processing, video processing – nearly any problem where there is a series of data elements

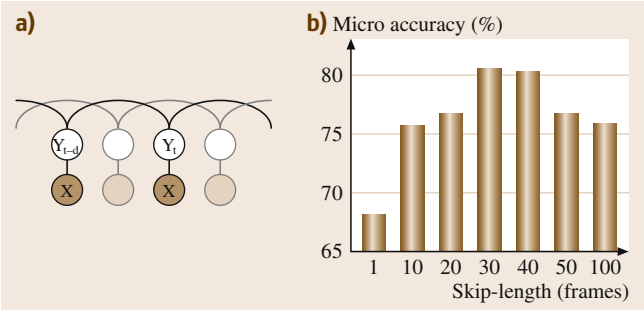


Fig. 5.15 **(a)** A skip chain CRF for inferring symbolic labels (Y) from robot kinematic and video data (X) acquired while a user performed a surgical training task. **(b)** The change in classification accuracy as a function of the skip length (images courtesy of Colin Lea)

from which some prediction is to be performed. For example, Fig. 5.15 shows the graphical structure of a skip-chain CRF designed to compute gesture labels from kinematic and video data acquired from a surgical robot [5.37]. This can be viewed as a discriminative generalization of a Hidden Markov Model (a generative model) that is designed to capture dependencies over a specified period of time (the *skip*). The right side shows the change in labeling performance as a function of the skip length which is now a *tunable* parameter of the model.

An in-depth discussion of CRFs goes well beyond this chapter. The interested reader is referred to [5.38, 39] to learn more about the underlying theory and application of CRFs. Because of their high interest, there are a number of open-source packages for developing and applying CRFs including PyStruct [5.40] for Python and CRF++ for C++.

5.4.3 Robust Estimation Methods

In our previous discussions, we generally assumed that all of the data was *good*, meaning that it was perhaps corrupted by noise but ultimately carried information about the problem at hand. However, in many cases, the data may contain so-called *outliers* – data points that are either much more highly corrupted than typical data, or which are completely spurious. For example, in our mapping application we might occasionally obtain range data through multiple reflections. Thus, while scanning a straight wall, most of the points would lie on a straight line, but occasionally we would have a data point that has a completely inconsistent range value.

Many common estimation methods are quite sensitive to data outliers. Consider a very simple case: estimating a single scalar value x by averaging a series of observations $X_1, X_2, \ldots X_N$. Then we can write our

Avidbots Ex1016
Page 149 of 2154

estimate $\hat{x}$ as

$$\hat{x} = \sum_{i=1}^{N} X_i / N \,. \tag{5.31}$$

Now, without loss of generality, suppose that X_N is an outlier. We can rewrite the above as

$$\hat{x} = \sum_{i=1}^{N-1} X_i / n + X_N / n \,. \tag{5.32}$$

It is now easy to see that we can produce *any* value of $\hat{x}$ by manipulating X_n. In short, a single outlier can create an arbitrarily poor estimate. More generally, the solution to any least-squares problem, e.g., estimating a line from laser range data, takes the general form $\hat{x} = \mathbf{M}y$. By the same argument as above, it is easy to show that any least-squares solution is likewise susceptible to outliers.

The field of *robust statistics* studies the problem of estimation or decision making when the underlying data are contaminated by outliers. In robust statistics, there are two important concepts: the *breakdown point* and *influence function*. The breakdown point is the proportion of outliers (i. e., data with arbitrarily large errors) that an estimator can tolerate before producing arbitrarily large errors in an estimate. We argued above that least-squares methods have a breakdown point of 0% since the estimate can be perturbed arbitrarily far by a single observation. By comparison, we might compute an estimate by taking the median of the data, which has a breakdown point of 50% – up to half of the data can be outliers and meaningful results may still be produced.

Whereas the breakdown point quantifies *how many* outliers can be tolerated, the influence function quantifies *how much* an outlier affects an estimate. In the case of least squares, the influence function is linear. One way of creating new estimators with better robustness is the method of *M-estimators* [5.21]. To produce an M-estimate, we consider the following minimization problem

$$\min_{\hat{x}} \sum_{i=1}^{N} \rho(\hat{x}, y_i) \,. \tag{5.33}$$

Note that defining $\rho(a, b) = (a - b)^2$ leads to a least-squares solution. However, we can now choose other functions with better resistance to outliers. Fig. 5.16 shows three common examples.

Note that, in general, the optimization of (5.33) is nonlinear and the result will often not exist in closed form. Interestingly, it is often possible to solve this problem using the method of *iteratively reweighted least squares* (IRLS) [5.30, 41]. The idea behind IRLS is quite simple. Recall that in (5.7) we introduced a weighting matrix $\mathbf{W}$. Suppose that, through some means, we knew which data points were outliers. In this case, we could simply set the weights for those points to zero, and the result would be the least-squares estimate on the remaining (good) data.

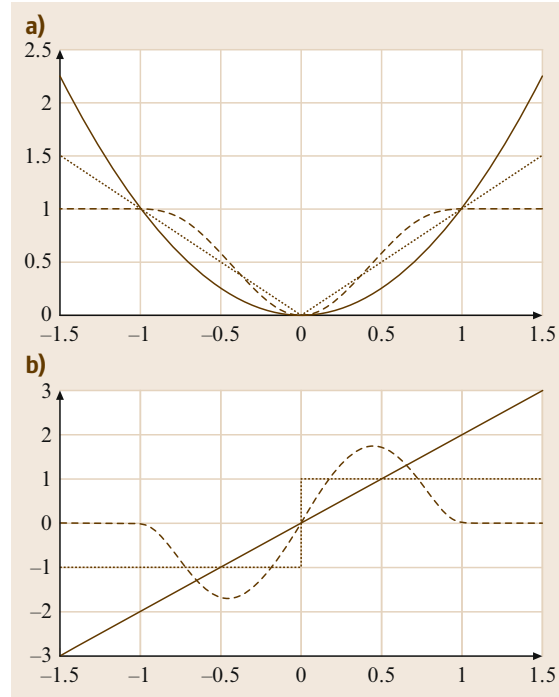


Fig. 5.16 **(a)** Three common robust M-estimation functions: the square function, the absolute value, and the Tukey biweight function. **(b)** The corresponding influence functions

In IRLS, we alternate between hypothesizing outliers (through reweighting) and solving to produce a solution (through least squares). Typically, the weight for a point depends on the residual error of the estimate. That is, suppose we compute

$$r = y - \mathbf{F}\hat{x} \,. \tag{5.34}$$

Let $\psi(y) = \mathrm{d}\rho / \mathrm{d}x \,|_{\hat{x}}$; then we can set $\mathbf{W}_{i,i} = \psi(y)/r_i$. It can be shown that in many cases this form of weighting will lead to convergence. An example of using IRLS techniques for video tracking is shown in Fig. 5.17.

Voting-Based Methods

Another common method for dealing with outliers is to choose a set of data and let it *vote* for a result. We dis-

Avidbots Ex1016
Page 150 of 2154

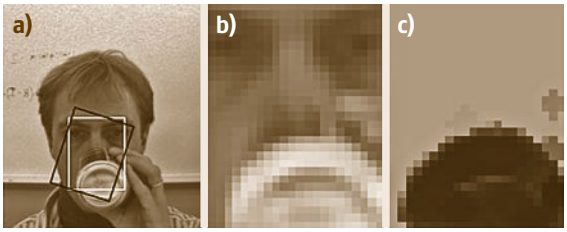


Fig.5.17a–c An example of using an M-estimate implemented via IRLS for visual tracking (after [5.42]). **(a)** Results of a face tracker in a single frame of video. The black frame corresponds to a tracking algorithm without outlier rejection and the white frame corresponds to the algorithm with outlier rejection. **(b)** Magnified view of the region in the white frame; **(c)** the corresponding weighting matrix in which darker areas mark outliers

cuss two common methods: RANSAC [5.11] and least median of squares (LMedS) [5.43].

In both cases, we start with the idea that, amidst all of the data (including outliers), there is an estimate that is consistent with the *good* data. The problem is to choose that estimate. Consider, however, our problem of estimating a line from laser data, and suppose we have 100 laser points. All we really need is to choose two points correctly, fit a line, and then count how many other points are consistent with this line. If we (conservatively) estimate that 3/4 of the data is good, then the odds of choosing two *good* points is 9/16, or equivalently, the odds of one or both points being outliers is 7/16. If we now repeat this process a few (e.g., ten) times, then the odds that *all* of our choices are bad is $(7/16)^{10} = 0.025\%$. To put it in other terms, there is a 99.975% chance we have chosen a *good* pair of points.

How do we decide to accept a sample? In RANSAC, we *vote* by counting the number of samples that are consistent with an estimate to within a given distance threshold. For example, we would choose points that are within a fixed distance to the line we estimated. We choose the candidate estimate with the largest number of votes. In LMedS, we instead compute the median distance of all of the samples to the line. We then choose the estimate with the least median value.

It is not hard to see that LMedS has a breakdown point of 50% of the data. RANSAC, on the other hand, can have a breakdown point that is potentially larger, but it requires the choice of a threshold. RANSAC also has the advantage that, once the inliers are identified, it is possible to compute a least-squares estimate from them, thus reducing the noise in the estimate.

Both RANSAC and LMedS can also provide good starting solutions for a robust iterative method such as IRLS.

5.4.4 Data Association Techniques

The previous section considered the case where there is a *known* relationship between observations and a quantity to be estimated. However, as was illustrated in our initial mobile robot mapping problem, it may be the case that we also have to compute this correspondence in conjunction with estimation. In this case, an essential step in estimation is the *data association* problem: producing a correspondence between the observed data and quantities to be estimated.

The literature on this problem is enormous; here we will focus on a few specific methods that have found wide use. We will also separate our discussion into *causal* (or sequential) association methods commonly used when filtering time-series data and *noncausal* (or batch) methods that can be used when the complete data set is available for processing. The latter is typically treated with methods for data *clustering*.

In both cases, we can extend our previous models and notation to include uncertainty as to the underlying source of the data. To this end, we will use a superscript on quantities to denote the observation model. Thus, our observation model becomes

$$x_{t+1}^k = g^k(x_t^k) + w_t^k \,, \tag{5.35}$$

$$y_t^k = f_t^k(x_t^k) + \eta_t^k \,, \tag{5.36}$$

where $k = 1 \ldots M$.

Clustering on Batch Data

Following the same course as our previous discussion on point estimation, let us first consider the case where we do not make any statistical assumptions about the data, and we have no system dynamics. Thus, we are simply given the observations $y_1, y_2, \ldots, y_M$. We have unknown underlying parameters $x^1, x^2, \ldots, x^N$ (for the moment, we take N as known). Our goal it to compute an *association mapping* π such that $\pi(j) = k$ if and only if y_j arose from the model parameters x^k.

k-Means Clustering

The k-means algorithm for clustering and data association is simple, well established, and forms a good starting point for our discussion. Here, we assume that $f(x) = x$ – that is, we are provided with noisy observations of the underlying state vectors. The k-means algorithm then proceeds as follows:

1. Pick N cluster centers $\{\hat{x}^i\}$.
2. For each observation y_j, associate it with the closest cluster center, that is, set $\pi(j) = i$, where

$$d(\hat{x}^i, y_j) = \min_k d(\hat{x}^k, y_j) \tag{5.37}$$

Avidbots Ex1016
Page 151 of 2154

for some distance function d (typically the Euclidean distance).

3. Estimate the mean of the observation associated with each cluster center as

$$\hat{x}^i = \sum_{j,\pi(j)=i} y_j \,. \tag{5.38}$$

4. Repeat steps 2 and 3.

In many cases and with good initialization, k-means works quite well. However, it can also fail to produce good clusters, and there is no guarantee that it will even converge to a solution. It is common to repeat the algorithm several times from different initial conditions and take the result that has the *best* outcome. Note also that the extension to linear observation models is straightforward by including $\mathbf{F}$ in (5.3) by defining

$$d(\hat{x}^i, y_j) = \|\mathbf{F}\hat{x}^i - y_j\| \tag{5.39}$$

and replacing (5.38) with the corresponding least-squares estimator. Going a step further, if we have a statistical model for observed data, then we could make use of the likelihood function introduced earlier and define $d(\hat{x}^i, y_j) = p(y_j|\hat{x}^i)$ and make use of the MLE in (5.38).

One disadvantage of the k-means algorithm is that, even when we have known statistical models, it is not guaranteed to converge. However, a variation, known as *expectation maximization*, can be shown to converge.

Expectation Maximization for Data Association and Modeling

The expectation-maximization (EM) algorithm [5.44] is a general statistical technique for dealing with missing data. In previous discussion, we made use of maximum-likelihood estimation to maximize the conditional probability of observed data given a set of unknown parameters. However, our use of MLE presumed that we had complete knowledge of the data. In particular, we knew the association between the data elements and models.

Let use now assume that some of our data is missing. To this end, define $\mathcal{Y}_\mathrm{O}$ and $\mathcal{Y}_\mathrm{U}$ as the *observed* and *unobserved* data, respectively. We then note that we can write

$$p(\mathcal{Y}_\mathrm{O}, \mathcal{Y}_\mathrm{U}|x) = p(\mathcal{Y}_\mathrm{U}|\mathcal{Y}_\mathrm{O}, x)p(\mathcal{Y}_\mathrm{O}|x) \,. \tag{5.40}$$

Suppose now that we make a guess for $\hat{x}$, and we have a distribution over the unknown data $\mathcal{Y}_\mathrm{U}$ (where this comes from we will discuss in a minute). It follows that we could compute the *expected value* of the log-likelihood function (recall that maximizing the log likelihood is equivalent to maximizing the likelihood) as

$$Q(x, \hat{x}) = E_{\mathcal{Y}_\mathrm{U}}\left[\log p(\mathcal{Y}_\mathrm{O}, \mathcal{Y}_\mathrm{U}|x)|\mathcal{Y}_\mathrm{O}, \hat{x}\right] \,. \tag{5.41}$$

Note that we differentiate between the fixed value $\hat{x}$ that is usually needed to define the distribution over the unknown data and the unknown x of the log-likelihood function.

Ideally, we would then like to choose values for x that make Q large. Thus, we can choose a new value according to the iterative rule

$$\hat{x}_i = \arg\max_x Q(x, \hat{x}_{i-1}) \,. \tag{5.42}$$

What can be shown is that this iteration will converge to some local maximum of the objective function Q. It is important to note that there is no guarantee that this is, however, the *global* maximum.

How do we connect this with clustering? We consider the observed data to be just that, the data we have observed. Let the unobserved data be the *association values* $\pi(j)$, $j = 1, 2, \ldots M$ that determine which model the observed data items originate from. Note that this is a discrete random variable. Let us further assume that N underlying clusters are distributed according to a Gaussian distribution with mean x_i and covariance Λ_i. Let the unconditional probability that a particular data item y_j comes from cluster i be α_i. The unknown parameters are then $\theta = \{x_1, x_2, \ldots, x_N, \Lambda_1, \Lambda_2, \ldots, \Lambda_N, \alpha_1, \alpha_2, \ldots, \alpha_N\}$. We now use $^-$ and $^+$ to denote prior and updated parameter estimates, respectively. For conciseness, we also define $w_{i,j} = p(\pi_j = i|y_j, \theta)$ and we use a superscript $^+$ to denote updated parameter estimates. Then, after a series of calculations [5.44], the EM algorithm for data clustering becomes

E-Step:

$$w_{i,j} = \frac{p(y_j|\pi(j) = i, \theta)\alpha_i}{\sum_i p(y_j|\pi(j) = i, \theta)\alpha_i} \,. \tag{5.43}$$

M-Step:

$$\hat{x}_i^+ = \frac{\sum_j y_j w_{i,j}}{\sum_j w_{i,j}} \,, \tag{5.44}$$

$$\Lambda_i^+ = \frac{\sum_j y_j (y_j)^t w_{i,j}}{\sum_j w_{i,j}} \,, \tag{5.45}$$

$$\alpha_i^+ = \frac{\sum_j w_{i,j}}{\sum_i \sum_j w_{i,j}} \,. \tag{5.46}$$

From this, we can see that EM produces a type of soft clustering, as opposed to k-means which produces

Avidbots Ex1016
Page 152 of 2154

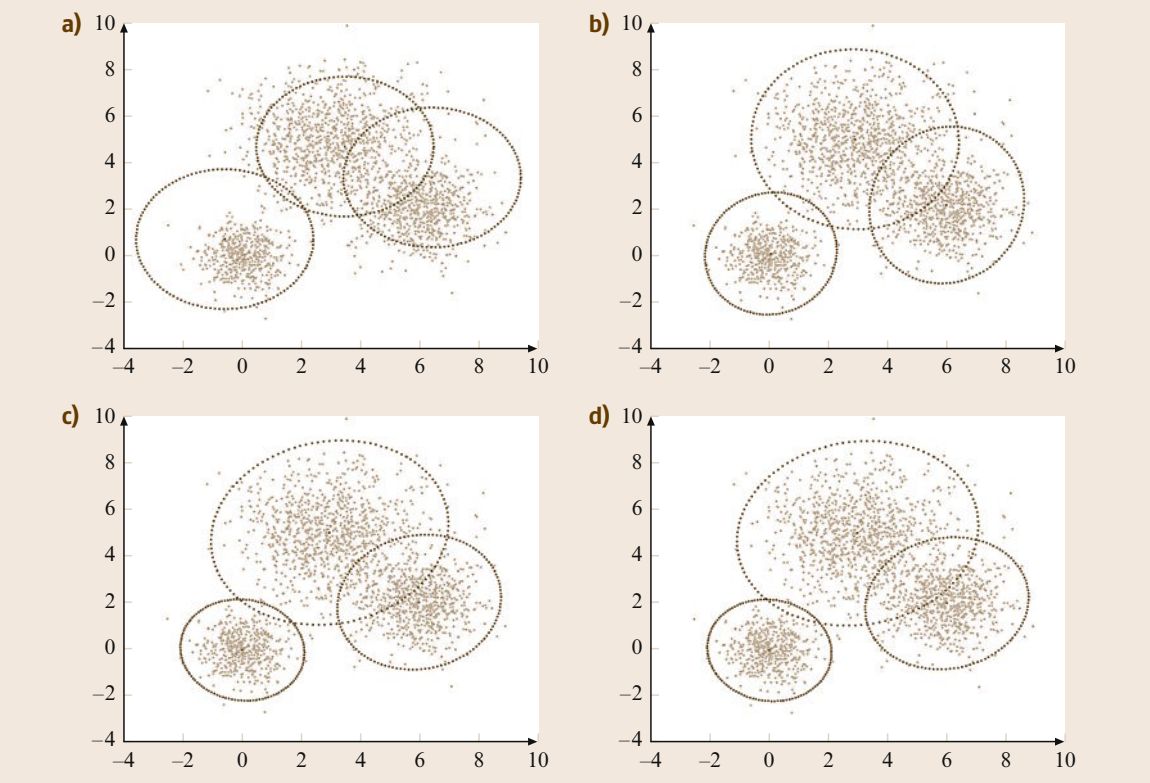


Fig.5.18a–d An example of clustering using expectation maximization. The figures are the results at iterations **(a)** 1, **(b)** 2, **(c)** 5, and **(d)** 10

specific decisions (in terms of $w_{i,j}$) as to which cluster an observation belongs. In fact, the result of estimation is the maximum-likelihood estimate of a *Gaussian mixture model* which has the form

$$p(y|\theta) = \sum_j \alpha_j N(y|\hat{x}_j, \mathbf{\Lambda}_j) \,, \tag{5.47}$$

where $N(\cdot)$ denotes a Gaussian density function. Fig. 5.18 shows the results of executing the EM algorithm on data sampled from a Gaussian mixture model.

Recursive Filtering

In the batch methods described above, we do not have a priori information on state parameters. In the case of recursive filtering, we have the advantage that prior state estimates, $\hat{x}_t^k$ and Λ_t^k, are available for processing at time $t+1$. As before, for data $y_t^i, i = 1 \ldots N$, the problem is to determine a mapping $\pi : \{1 \ldots N\} \rightarrow \{1 \ldots M\}$ which *associates* data element i to model $k = \pi(i)$. In some cases, it is also useful to include an *outlier process* to handle data that comes from no known model. For this purpose, we can include 0 in the range of the function, and use the mapping to zero as an outlier.

Nearest-Neighbor Association

Analogous to k-mean clustering, a simple way of producing a data association is to compute the data association value as

$$\pi(i) = \arg\min_j d(\mathbf{F}^j \hat{x}^j, \hat{y}^i) \,. \tag{5.48}$$

However, nearest-neighbor methods do not take into account what we know about either the sensor data or the estimate. That is, we may have a very very good estimate of some model i and a very very bad estimate for some other model j. If a sensor observation is equidistance between them, does it make sense to flip a coin? Odds are that it is more likely to come from j (with a larger variance) than i (with a smaller variance).

A commonly used measure that can take this into account is the *Mahalanobis* distance [5.45]. The idea is to weight each value by its variance as

$$m(y_1, y_2) = (y_1 - y_2)(\mathbf{\Lambda}_1 + \mathbf{\Lambda}_2)^{-1}(y_1 - y_2)^{\mathrm{T}} \,. \tag{5.49}$$

Thus, distances are scaled inversely with uncertainty. In the case above, the observation with a higher variance would produce the smaller distance, as desired.

Avidbots Ex1016
Page 153 of 2154

Even using this as a weighting method, it is still possible that we will make an error in data association. From an estimation point of view, this will introduce an outlier in the estimation process with, as discussed above, potentially disastrous results. Another approach, analogous to IRLS, is instead to weight the data based on the distance to a model. This leads naturally to the notion of a *data association filter*. We refer the reader to [5.20] for extensive discussions of these techniques.

5.4.5 Modeling Sensors

To this point, we have introduced several sensing modalities, and we have discussed several methods for estimation. However, the latter often rely on having statistical models for the former. Thus, no chapter on sensing and estimation would be complete without a short discussion of modeling sensors.

Developing a sensor model potentially involves four major elements:

1. Creating a physical model
2. Determining a sensor calibration
3. Determining an error model
4. Identifying failure conditions.

The physical model is the relationship f between the underlying quantities of interest (x) and the available data (y). In many cases, this relationship is obvious, e.g., the distance from a laser sensor to a surface in the world. In others, it may be less so, e.g., what is the right model relating intensities in multiple camera images to the distance to an observed point? In some cases, it may be necessary to include computational processes, e.g., feature detection and correspondence, in the sensor model.

Once a physical model is determined, there is often a process of sensor calibration. Such procedures are typically specific to the sensor in question, for example, the imaging geometry of a perspective camera system requires identification of two scale parameters (governing image scale) and the location of the optical center (two additional parameters). There are also often lens distortion parameters. These parameters can only be determined by a careful calibration procedure [5.7].

Once a calibrated physical sensor model is available, determining an error model typically involves performing an identification of the statistical parameters. Ideally, the first step is to determine an empirical distribution on errors. However, this can often be difficult, as it requires knowing accurate *ground truth* for the underlying unknown parameters. This often requires the development of a laboratory setup that can simulate the expected sensing situation.

Given such an empirical distribution, there are several important questions, including:

1. Are observations statistically independent?
2. Is the error distribution unimodal?
3. Can the essential aspects of the empirical error be captured using common statistical quantities such as the data variance?

We refer the reader to books on statistics and data modeling [5.46] for further information on this topic.

Finally, it is important to understand when sensors can and cannot provide reliable data, for example, a laser sensor may be less accurate on dark surfaces than on light ones, cameras do not produce meaningful data if the lighting is too bright or too dark, and so forth. In some cases, there are simple clues to these conditions (e.g., simply looking at the intensity histogram of a camera image can quickly determine if conditions are suitable for processing). In some cases it is only possible to detect conditions in context (e.g., two range sensors disagree on the distance to a surface). In some cases failure is only detectable in retrospect, e.g., after a 3-D surface model is built it is apparent that a hypothesized surface would be occluded by another and must therefore be a multiple reflection. In a truly robust sensing system, all available possibilities for verifying sensor operation should be exploited.

5.4.6 Other Uncertainty Management Methods

Due to the limitations of space, we have necessarily limited our discussion to cover the most commonly used sensing and estimation methods. It is important to note that many other alternative uncertainty management methods have been proposed and employed with success.

For example, if it is known that sensing error is bounded, constraint-based methods can be quite effective at performing point estimation [5.47, 48]. Alternatively, if only partial probability models can be identified, Dempster–Shafer methods can be employed to make judgments [5.49].

Fuzzy logic allows graded membership of a set. With fuzzy set theory it is possible to have partial membership. As an example in classification of data it might be difficult to select between two categories such as *average* and *tall* and gradual shifts may make sense. Such methods have for example been used for situation assessment and navigation as reported by [5.50] for the DAMN architecture.

Avidbots Ex1016
Page 154 of 2154

5.5 Representations

Sensor data can be used directly for control but it is also used for estimation of the state of the robot and/or the world. The definition of *state* and the appropriate methods for estimation are closely related to the representation adopted for the application.

There are a rich variety of possible world representations including most typical geometric elements such as points, curves, surfaces, and volumes. A fundamental aspect in robotics is the concept of rigid-body *pose*. The pose of a robot or an entity in the world is characterized by position and orientation with respect to a reference frame.

In general, pose is represented by the pair $(\mathbf{R}, \boldsymbol{H})$. Here $\mathbf{R}$ is the orientation of the object represented by a rotation matrix with respect to a reference frame. Similarly, $\boldsymbol{H}$ represents the translation of the object with respect the reference frame. There is a rich set of potential representations for the transformation between reference frames as detailed in the chapter on Kinematics (Chap. 2) and in [5.51].

Sensory data is acquired in a local sensor reference frame, for example, a sonar transducer, a laser scanner, and a stereo imaging system would all measure distances to surfaces in the world relative to their own frame. However, if the goal is to combine this information into a common world model, the data must be transformed into a robot-centered reference frame, or possibly into a fixed world (inertial) reference frame. In particular, the world-centered reference frame enable simple transfer across robot motions and communication to other robots and/or users.

For the purposes of discussion, most representations for the integration of sensor data can be categorized into four general classes of models:

- Raw sensor data models
- Grid-based models
- Feature-based models
- Symbolic or graphical models.

Naturally, it is also possible to combine elements of these four categories to achieve hybrid models of the environment.

5.5.1 Raw Sensor Representations

For simple feedback control [5.52] it is common to integrate raw sensory data directly into the control system, as in many cases it is unnecessary to have a *world model* for the control. For example, proprioceptive sensing is often used in this manner: basic trajectory control makes direct use of encoder information from joints, and force control operates directly from force or torque information from force sensors.

Raw sensor models are less common with exteroceptive sensing, but there are cases where it can be useful. One example is mobile robot mapping from dense point data. This approach has in particular been made popular for laser range sensors, where scan alignment is used for the generation of point-based world models. The work by [5.53, 54] demonstrates how a number of laser range scans can be combined into a joint model of the environment. More formally a scan of the environment at time t is represented as a point set

$$\mathcal{P}_t = \{p_i = (\rho_i, \theta_i) | i \in 1 \ldots N\} \,. \tag{5.50}$$

Two different scans $\mathcal{P}_t$ and $\mathcal{P}_{t+1}$ are then aligned through a standard rigid body transformation. The estimation of the transformation is typically achieved through use of the ICP algorithm [5.15]: assume that $\boldsymbol{H}^{[0]}$ is an initial estimate of the transformation between the two point sets and that $||\boldsymbol{p}_t - \boldsymbol{p}_{t+1}||$ is the Euclidean distance between a point from $\mathcal{P}_t$ and a point from $\mathcal{P}_{t+1}$. If furthermore CP is a function to locate the closest point from one set in the other set, then let C be the set of point correspondences between the two sets. Through iterations of the following algorithm,

1. Compute $C_k = \cup_{i=1}^{N} \{\boldsymbol{p}_i, CP[\boldsymbol{H}^{[k-1]}(\boldsymbol{p}_i, \mathcal{P}_{t+1})]\}$,
2. Estimate the $\boldsymbol{H}^{[k]}$ that minimizes the LSQ error between the points in C_k until the error has converged

an estimate of the scan alignment can be found and a joint model of the environment can be constructed.

The model is simple to construct and well suited for integration of sensor data from a single modality. Typically the model does not include information about uncertainty and, as the model grows the complexity, $O(\sum_t |\mathcal{P}_t|)$ becomes an issue.

5.5.2 Grid-Based Representations

In a grid-based representation the world is tessellated into a number cells. The cells can contain information about environmental features such as temperature, obstacles, force distribution, etc. The dimensionality of the grid is typically two or three, depending on the application. The tessellation can either be uniform or tree based using quad-tree or oct-trees [5.55]. The tree-based methods are in particular well suited for handling of inhomogeneous and large-scale data sets. In a grid model each cell contains a probability over the parameter set. As an example, when using

Avidbots Ex1016
Page 155 of 2154

the grid model for representation of a physical environment, the cell specifies *occupied* (O) or *free* (F) and the cell encodes the probability P(occupied). Initially where there is no information the grid is initialized to $P(\mathrm{O}) = 0.5$ to indicate unknown. It is further assumed that sensor models are available that specify $P(R|S_{ij})$, i. e., the probability of detection objects for a given sensor and location. Using Bayes theorem (5.10) it is now possible to update the grid model according to

$$p_{ij}(t+1) = \frac{P(R|S_{ij}=O)p_{ij}(t)}{P(R|S_{ij}=O)p_{ij}(t) + P(R|S_{ij}=F)(1-p_{ij}(t))} ,$$

where p_{ij} is computed across the grid model whenever new data are acquired.

The grid-based model has been widely used in mobile robotics [5.56, 57] and in medical imaging where image volumes are quite common [5.58]. Volume models can be relative large. As an example a millimeter-resolution grid model of the human head requires 4 GB of storage, and thus demands significant computational resources for maintenance.

5.5.3 Feature Representations

Both the raw sensor representation and the grid-based models contain a minimum of abstraction for the sensory data. In many cases there is an interest in extracting features from the sensor data to reduce the storage requirement and only preserve data that are invariant across motion of the platform or external objects. Features span most standard geometric entities such as points ($\boldsymbol{p}$), lines ($\boldsymbol{l}$), planes ($\boldsymbol{N}, \boldsymbol{p}$), curves ($\boldsymbol{p}(s)$), and more general surfaces. For estimation of properties of the external world there is a need for a hybrid model in which collections of features are integrated into a unified model of state.

In general a point is represented in $\mathcal{R}(3)$. Sensors have associated noise and, consequently, in most cases points have an associated uncertainty, typically modeled as Gaussian with mean μ and standard deviation σ. The estimation of the statistics is achieved using first- and second-order moments.

Line features are more difficult to represent. The mathematical line can be represented by the vector pair $(\boldsymbol{p}, \boldsymbol{t})$, i. e., a point on the line and the tangent vector. In many practical applications the line has a finite extent, and there is a need to encode the length of the line, which can be achieved using end points, start point, tangent, and length. In some cases it is advantageous to have a redundant representation of the line model to simplify updating and matching. The relation between end-point uncertainties and other line parameters can be derived analytically, as described in [5.59]. The estimation of line parameters is often based on the previously describe RANSAC method through the use of the *Hough* transform [5.10], which is another voting-based method.

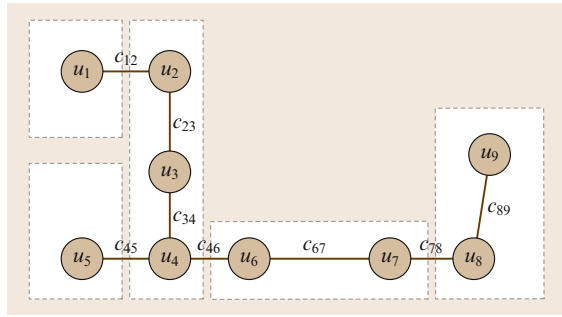


Fig. 5.19 A topological map of a spatial environment

For more complex feature models such as curves or surfaces there is a corresponding need to utilize detection methods that facilitate robust segmentation of features, and estimation of the associated uncertainty. A comprehensive description of such methods is available from [5.44].

5.5.4 Symbolic/Graph-Based Models

All of the representations presented in Sects. 5.5.1–5.5.3 are parametric in nature with limited associated semantics. Methods for the recognition of structures, spaces, locations, and objects have seen major recent progress in particular due to advances in statistical learning theory [5.12, 60]. Consequently, today there exist a variety of methods for the recognition of complex structures in sensor data, such as landmarks, road surfaces, body structures, etc. Given the availability of recognized structures it is possible to represent the environment using the previously discussed graphical models. In general a graph is composed of a set of nodes N and a set of edges E that connect nodes. Both nodes and edges can have attributes associated such as labels and distances. One example of a graph structure is a topological map of the environment as shown in Fig. 5.19. The graph representation could also be a semantic model of the environment (objects and places) or a representation of the composition of an object to assembled.

In terms of model updating semantic/graph-based representations can take advantage of recent advances in Bayesian reasoning as presented by *Pearl* [5.61], and exemplified in [5.62].

Avidbots Ex1016
Page 156 of 2154

5.6 Conclusions and Further Readings

Sensing and estimation continues to be a challenging and very active area of robotics research. Several areas of sensing such as computer vision and medical imaging are themselves large and diverse research areas. At the same time, new fundamental and applied techniques in estimation continue to be developed. Indeed, it is fair to say that perception continues to be one of the most challenging areas of robotics research.

Given this wealth of activity, no single chapter can hope to cover all of the material that can be useful in the development of sensor-based robotics. However, the methods that have been presented here are representative of the most commonly used techniques in robotics. In particular, linear techniques such as the Kalman filter continue to form the backbone of perceptive robotics. Part C of the handbook provides more in-depth coverage of several of the key topics in sensing and estimation.

For the reader wishing to learn more, general discussion on the design, physics, and use of a rich variety of sensors can be found in the Handbook of Modern Sensors [5.3]. A discussion of sensors for mobile robots can be found in [5.63], though significant advances have been achieved since the book was published more than a decade ago. Sensing and estimation using computer vision is described in detail in [5.64] and [5.65].

The basic estimation theory is covered in a number of excellent text books. Much of the detection and linear estimation theory is covered in depth in [5.20] and [5.66]. General statistical estimation is covered in [5.12] and [5.13] and the more recently updated version [5.44]. Robust methods are described in detail in [5.21, 43]. In-depth coverage of estimation methods for mobile systems is also covered in [5.33].

References

5.1 D.C. Marr: *Vision* (Freeman, Bedford 1982)
5.2 R. Siegwart, I.R. Nourbakhsh, D. Scaramuzza: *Introduction to Autonomous Mobile Robots*, Intelligent Robotics and Autonomous Systems (MIT Press, Cambridge 2011)
5.3 J. Fraden: *Handbook of Modern Sensors: Physic, Design and Applications*, 2nd edn. (Springer, New York 1996)
5.4 H. Yousef, M. Boukallel, K. Althoefer: Tactile sensing for dexterous in-hand manipulation in robotics – A review, Sensors Actuators A: Physical **167**(2), 171–187 (2011)
5.5 M.I. Tiwana, S.J. Redmond, N.H. Lovell: A review of tactile sensing technologies with applications in biomedical engineering, Sensors Actuators A: Physical **179**, 17–31 (2012)
5.6 G. Dissanayaka, S. Sukkarieh, E. Nebot, H. Durrant-Whyte: The aiding of a low-cost strapdown inertial measurement unit using vehicle model constraints for land vehicle applications, IEEE Trans. Robot. Autom. **17**(5), 731–748 (2001)
5.7 Z. Zhang: A flexible new technique for camera calibration, IEEE Trans. Pattern. Anal. Mach. Intell. **22**(11), 1330–1334 (2000)
5.8 D. Burschka, J. Geiman, G.D. Hager: Optimal landmark configuration for vision-based control of mobile robots, Proc. Int. Conf. Robot. Autom. ICRA (2003) pp. 3917–3922
5.9 J. Baker, N. Anderson, P. Pilles: Ground-penetrating radar surveying in support of archeological site investigations, Comput. Geosci. **23**(10), 1093–1099 (1997)
5.10 P.V.C. Hough: A method and means for recognizing complex patterns, U.S. Patent 306 9654 (1962)
5.11 M.A. Fischler, R.C. Bolles: Random Sample concensus: A paradigm for model fitting with applications to image analysis and automated cartography, Commun. ACM **24**, 381–395 (1981)
5.12 T. Hastie, R. Tibshirani, J. Friedman: *The Elements of Statistical Learning*, Springer Series in Statistics (Springer, Berlin, Heidelberg 2002)
5.13 R.O. Duda, P.E. Hart: *Pattern Classification and Scene Analysis* (Wiley-Interscience, New York 1973)
5.14 R. Vidal, Y. Ma, J. Piazzi: A new GPCA algorithm for clustering subspaces by fitting, differentiating and dividing polynomials, Proc. Int. Conf. Cumput. Vis. Pattern Recog. **1**, 510–517 (2004)
5.15 P. Besl, N.D. McKay: A method for registration of 3-D shapes, IEEE Trans. Pattern Anal. Mach. Intell. **14**(2), 239–256 (1992)
5.16 F. Dellaert, S. Seitz, C. Thorpe, S. Thrun: Special issue on Markov chain Monte Carlo methods, Mach. Learn. **50**, 45–71 (2003)
5.17 A. Gelb (Ed.): *Applied Optimal Estimation* (MIT Press, Cambridge 1974)
5.18 D. Simon: *Optimal State Estimation: Kalman, H Infinity, and Nonlinear Approaches* (Wiley, New York 2006)
5.19 A. Doucet, N. de Freitas, N. Gordon: *Sequential Monte Carlo Methods in Practice* (Springer, Berlin, Heidelberg 2001)
5.20 Y. Bar-Shalom, T. Fortmann: *Tracking and Data Association* (Academic, New York 1988)
5.21 P.J. Huber: *Robust Statistics* (Wiley, New York 1981)
5.22 J.O. Berger: *Statistical Decision Theory and Bayesian Analysis*, 2nd edn. (Springer, New York 1985)
5.23 G.D. Hager: *Task-Directed Sensor Fusion and Planning* (Kluwer, Boston 1990)

Avidbots Ex1016
Page 157 of 2154

5.24 S. Abrams, P.K. Allen, K. Tarabanis: Computing camera viewpoints in a robot work-cell, Int. J. Robot. Res. **18**(3), 267–285 (1999)
5.25 M. Suppa, P. Wang, K. Gupta, G. Hirzinger: *C-space exploration using noisy sensor models*, *Proc* (IEEE, Int. Conf. Robot. Autom 2004) pp. 1927–1932
5.26 G.S. Chirikjian, A.B. Kyatkin: *Engineering Applications of Noncommutative Harmonic Analysis* (CRC, Boca Raton 2000)
5.27 J.C. Kinsey, L.L. Whitcomb: Adaptive identification on the group of rigid body rotations and its application to precision underwater robot navigation, IEEE Trans. Robot. **23**, 124–136 (2007)
5.28 P.J. Bickel, K.A. Doksum: *Mathematical Statistics*, 2nd edn. (Prentice-Hall, Upper Saddle River 2006)
5.29 G. Strang: *Linear Algebra and its Applications*, 4th edn. (Brooks Cole, New York 2005)
5.30 P. McCullagh, J.A. Nelder: *Generalized Linear Models*, 2nd edn. (Chapman Hall, New York 1989)
5.31 E.L. Lehmann, G. Casella: *Theory of Point Estimation* (Springer, New York 1998)
5.32 R.E. Kalman: A new approach to linear filtering and prediction problems, Transactions of the ASME, J. Basic Eng. **82**, 35–45 (1960)
5.33 S. Thrun, D. Fox, W. Burgard: *Probabilistic Robotics, Autonomous Robotics and Intelligent Agents* (MIT Press, Cambridge 2005)
5.34 C. Bishop: *Pattern Recognition and Machine Learning* (Springer, New York 2006)
5.35 T. Joachims, T. Finley, C.-N.J. Yu: Cutting-plane training of structural SVMs, Mach. Learn. **77**(1), 27–59 (2009)
5.36 S. Lacoste-Julien, M. Jaggi, M. Schmidt, P. Pletscher: Block-coordinate Frank-Wolfe optimization for structural SVMs, Proc. Int. Conf. Mach. Learn. (2013) pp. 53–61
5.37 L. Tao, L. Zappella, G.D. Hager, R. Vidal: Surgical gesture segmentation and recognition, Med. Image Comput. Comput.-Assisted Intervent., MICCAI 2013 (2013) pp. 339–346
5.38 C. Sutton, A. McCallum: An introduction to conditional random fields for relational learning. In: *Introduction to Statistical Relational Learning*, ed. by L. Getoor, B. Taskar (MIT Press, Cambridge 2006) pp. 93–128
5.39 C. Sutton, A. McCallum: An introduction to conditional random fields, Found. Trends Mach. Learn. **1**, 2055–2060 (2010)
5.40 C.A. Müller, S. Behnke: PyStruct-learning structured prediction in python, J. Mach. Learn. Res. **1**, 2055–2060 (2013)
5.41 J.W. Hardin, J.M. Hilbe: *Generalized Linear Models and Extensions*, 2nd edn. (Stata, College Station 2007)
5.42 G.D. Hager, P.N. Belhumeur: Efficient region tracking of with parametric models of illumination and geometry, IEEE Trans. Pattern Anal. Mach. Intell. **20**(10), 1025–1039 (1998)
5.43 P.J. Rousseauw, A. Leroy: *Robust Regression and Outlier Detection* (Wiley, New York 1987)
5.44 R.O. Duda, P.E. Hart, D.G. Stork: *Pattern Classification*, 2nd edn. (Wiley, New York 2001)
5.45 P.C. Mahalanobis: On the generalised distance in statistics, Proc. Nat. Inst. Sci. India **12**, 49–55 (1936)
5.46 J. Hamilton: *Time Series Analysis* (Princeton Univ. Press, Princeton 1994)
5.47 S. Atiya, G.D. Hager: Real-time vision-based robot localization, IEEE Trans. Robot. Autom. **9**(6), 785–800 (1993)
5.48 G.D. Hager: Task-directed computation of qualitative decisions from sensor data, IEEE Trans. Robot. Autom. **10**(4), 415–429 (1994)
5.49 G. Shafer: *A Mathematical Theory of Evidence* (Princeton Univ. Press, Princeton 1976)
5.50 J. Rosenblatt: *DAMN: A distributed architecture for mobile navigation*, *AAAI 1995* (Spring, Symposium on Lessons Learned for Implementing Software Architectures for Physical Agents 1995) pp. 167–178
5.51 R.M. Murrey, Z. Li, S. Sastry: *A Mathematical Introduction to Robotic Manipulation* (CRC, Boca Raton 1993)
5.52 K.J. Åström, B. Wittenmark: *Adaptive Control*, 2nd edn. (Addison-Wesley, Reading 1995)
5.53 S. Gutmann, C. Schlegel: AMOS: Comparison of scan-matching approaches for self-localization in indoor environments, 1st Euromicro Conf. Adv. Mobile Robotics (1996) pp. 61–67
5.54 S. Gutmann: *Robust Navigation for Autonomous Mobile Systems*, Ph.D. Thesis (Alfred Ludwig University, Freiburg 2000)
5.55 H. Samet: The quadtree and related hierarchical data structures, ACM Comput. Surv. **16**(2), 187–260 (1984)
5.56 A. Elfes: Sonar-based real-world mapping and navigation, IEEE Trans. Robot. Autom. **3**(3), 249–265 (1987)
5.57 A. Elfes: *A Probabilistic Framework for Robot Perception and Navigation*, Ph.D. Thesis (Carnegie Mellon University, Pittsburgh 1989)
5.58 M.R. Stytz, G. Frieder, O. Frieder: Three-dimensional medical imaging: Algorithms and computer systems, ACM Comput. Surv. **23**(4), 421–499 (1991)
5.59 R. Deriche, R. Vaillant, O. Faugeras: From Noisy Edges Points to 3D Reconstruction of a Scene: A robust approach and its uncertainty analysis. In: *Theory and Applications of Image Analysis*, (World Scientific, Singapore 1992) pp. 71–79
5.60 V.N. Vapnik: *Statistical Learning Theory* (Wiley, New York 1998)
5.61 J. Pearl: *Probabilistic Reasoning in Intelligent Systems* (Morgan Kaufmann, New York 1988)
5.62 M. Paskin: *Thin Junction Tree Filters for Simultaneous Localisation and Mapping*, Ph.D. Thesis (University of California, Berkley 2002)
5.63 H.R. Everett: *Sensors for Mobile Robots: Theory and Application* (Peters, London 1995)
5.64 D. Forsyth, J. Ponce: *Computer Vision – A Modern Approach* (Prentice-Hall, Upper Saddle River 2003)
5.65 R. Hartley, A. Zisserman: *Multiple View Geometry in Computer Vision* (Cambridge Univ. Press, Cambridge 2000)
5.66 S. Blackman, R. Popoli: *Design and Analysis of Modern Tracking Systems* (Artech House, London 1999)

Avidbots Ex1016
Page 158 of 2154

Multimedia Contents

6. Model Identification

John Hollerbach, Wisama Khalil, Maxime Gautier

This chapter discusses how to determine the kinematic parameters and the inertial parameters of robot manipulators. Both instances of model identification are cast into a common framework of least-squares parameter estimation, and are shown to have common numerical issues relating to the identifiability of parameters, adequacy of the measurement sets, and numerical robustness. These discussions are generic to any parameter estimation problem, and can be applied in other contexts.

For kinematic calibration, the main aim is to identify the geometric Denavit–Hartenberg (DH) parameters, although joint-based parameters relating to the sensing and transmission elements can also be identified. Endpoint sensing or endpoint constraints can provide equivalent calibration equations. By casting all calibration methods as closed-loop calibration, the calibration index categorizes methods in terms of how many equations per pose are generated.

Inertial parameters may be estimated through the execution of a trajectory while sensing one or more components of force/torque at a joint. Load estimation of a handheld object is simplest because of full mobility and full wrist force-torque sensing. For link inertial parameter estimation, restricted mobility of links nearer the base as well as sensing only the joint torque means that not all inertial parameters can be identified. Those that can be identified are those that affect joint torque, although they may appear in complicated linear combinations.

6.1 **Overview** 113
6.2 **Kinematic Calibration** 115
6.2.1 Serial-Link Robot Manipulators 116
6.2.2 Parallel Manipulator Calibration 120
6.3 **Inertial Parameter Estimation** 122
6.3.1 Link Inertial Parameter Estimation. 122
6.3.2 Load Inertial Parameter Estimation 124
6.3.3 Identification of Total Joint Drive Gains 126
6.3.4 Link Parameter Estimation for More Complex Structures 126
6.4 **Identifiability and Numerical Conditioning** 127
6.4.1 Identifiability 128
6.4.2 Observability 131
6.4.3 Scaling 132
6.4.4 Recursive Least Squares and the Kalman Filter 134
6.5 **Conclusions and Further Reading** 135
6.5.1 Relation to Other Chapters 135
6.5.2 Further Reading 136
Video-References 136
References 137

6.1 Overview

There are many different kinds of models in robotics, whose accurate identification is required for precise control. Examples from the previous chapters include sensor models, actuator models, kinematic models, dynamic models, and flexibility models. System identification is the general field concerned with the process of identifying models from measurements. Generally speaking, there are two types of models: parametric and nonparametric models. Parametric models are described by a few parameters, which are adequate to

characterize the accuracy of a model throughout the working range. Examples include sensor gain and offset, the DH parameters for links, and rigid-body inertial parameters. Parametric models are particularly appropriate for robotics, whose components are manmade and whose properties are controlled and understood.

Nonparametric models include the impulse response and the Bode plot for linear systems, and Wiener and Volterra kernels for nonlinear systems [6.1]. A nonparametric model can be used as a stepping stone towards identifying a parametric model; for example, a Bode plot (graph of amplitude and phase versus input frequency) is often used to decide on model order, such as whether an actuator should be modeled as a second- or third-order system. Otherwise, nonparametric models are required when the system's properties are so complicated that a few lumped parameters do not suffice. This is particularly true for biological systems.

This chapter will describe the parametric calibration of the following kinds of models:

1. Kinematic parameters. Kinematic calibration is the process of locating coordinate systems of objects relative to each other. These objects may be isolated from each other, or they may be linked by joints. Examples include:
   - Locating a robot relative to a global reference frame.
   - Locating a stereo vision system relative to a robot.
   - Locating a grasped object relative to the manipulator's gripper frame.
   - Locating neighboring coordinate systems of links of a robot manipulator.
2. Rigid-body inertial parameters. These parameters are required to predict the driving forces and torques to move objects and manipulators.

Suppose there are N_{par} parameters combined into an $N_{\text{par}} \times 1$ parameter vector $\boldsymbol{\phi} = \{\phi_1, \ldots, \phi_{N_{\text{par}}}\}$. The parameters may appear linearly or nonlinearly in the model.

$$\text{Linear model:} \qquad \boldsymbol{y}^l = \mathbf{A}^l \boldsymbol{\phi} \,, \tag{6.1}$$

$$\text{Nonlinear model:} \qquad \boldsymbol{y}^l = \boldsymbol{f}(\boldsymbol{x}^l, \boldsymbol{\phi}) \,, \tag{6.2}$$

where $\boldsymbol{y}^l = \{y_1^l, \ldots, y_M^l\}$ is the $M \times 1$ vector of output variables and $\boldsymbol{x}^l = \{x_1^l, \ldots, x_n^l\}$ is the vector of input variables. For the linear model, $\mathbf{A}^l$ is an $M \times N_{\text{par}}$ matrix whose entries A_{ij}^l are functions of the input variables $\boldsymbol{x}^l$. Any entry A_{ij}^l may be a complicated nonlinear function of $\boldsymbol{x}^l$, but it evaluates to just a number. For the nonlinear model, an explicit equation is shown in which the input variables appear in a nonlinear function $\boldsymbol{f} = \{f_1, \ldots, f_M\}$. Implicit nonlinear models $\boldsymbol{f}(\boldsymbol{y}^l, \boldsymbol{x}^l, \boldsymbol{\phi}) = \mathbf{0}$ may also appear in calibration [6.2]; they are handled similarly to explicit nonlinear models (Sect. 6.2.2). There may be P different measurements; a particular measurement is indicated by the superscript $l = 1, \ldots, P$.

For linear models, information from different measurements is combined by stacking the P (6.1)

$$\boldsymbol{y} = \mathbf{A}\boldsymbol{\phi} \,, \tag{6.3}$$

where $\boldsymbol{y} = \{\boldsymbol{y}^1, \ldots, \boldsymbol{y}^l\}$ is an $MP \times 1$ vector of all output measurements and $\mathbf{A} = \{\mathbf{A}^1, \ldots, \mathbf{A}^l\}$ is $MP \times N_{\text{par}}$. The parameters are estimated by ordinary least squares

$$\boldsymbol{\phi} = (\mathbf{A}^{\mathrm{T}}\mathbf{A})^{-1}\mathbf{A}^{\mathrm{T}}\boldsymbol{y} \,. \tag{6.4}$$

In statistics, the matrix $\mathbf{A}$ is called the regressor matrix and the least-squares solution is called regression [6.3]. An example of a linear model is the rigid-body model for inertial parameters.

The Gauss–Newton method [6.3] is typically employed to estimate the nonlinear model (6.2). First, absorb the input variables $\boldsymbol{x}^l$ (which can be considered as a number of constants) into the nonlinear function $\boldsymbol{f}^l$, now given a superscript. The model is linearized through a Taylor series expansion at a current estimate $\boldsymbol{\phi}^k$ of the parameters at iteration k

$$\begin{aligned} \boldsymbol{y}_{\text{c}}^l &= \boldsymbol{f}^l(\boldsymbol{\phi}^k + \Delta\boldsymbol{\phi}) \\ &= \boldsymbol{f}^l(\boldsymbol{\phi}^k) + \left.\frac{\partial \boldsymbol{f}^l(\boldsymbol{\phi})}{\partial \boldsymbol{\phi}}\right|_{\boldsymbol{\phi}=\boldsymbol{\phi}^k} \Delta\boldsymbol{\phi} \\ &\quad + \text{higher order terms} \\ &\approx \boldsymbol{f}^l(\boldsymbol{\phi}^k) + \mathbf{A}^l\Delta\boldsymbol{\phi} \,, \end{aligned} \tag{6.5}$$

where $\boldsymbol{y}_{\text{c}}^l$ are the computed values of the output variables and $\mathbf{A}^l = \partial \boldsymbol{f}^l / \partial \boldsymbol{\phi}$ is a Jacobian matrix evaluated at $\boldsymbol{\phi}^k$. Higher-order terms in the Taylor series are ignored, yielding the linearized form (6.5). The bold assumption is now made that a correction $\Delta\boldsymbol{\phi}$ to the parameter estimate $\boldsymbol{\phi}^k$ causes the computed output variables to equal the measurements: $\boldsymbol{y}_{\text{c}}^l = \boldsymbol{y}^l$. Defining $\Delta\boldsymbol{y}^l = \boldsymbol{y}^l - \boldsymbol{f}^l(\boldsymbol{\phi}^k)$ as the error between the output measurement and the predicted output with the current model $\boldsymbol{\phi}^k$, the linearized (6.5) becomes

$$\Delta\boldsymbol{y}^l = \mathbf{A}^l\Delta\boldsymbol{\phi} \,. \tag{6.6}$$

The linearized equation is then stacked for P measurements for the estimation form

$$\Delta\boldsymbol{y} = \mathbf{A}\Delta\boldsymbol{\phi} \,. \tag{6.7}$$

Avidbots Ex1016
Page 160 of 2154

A correction $\Delta\boldsymbol{\phi}$ to the parameter estimates is now found by ordinary least squares

$$\Delta\boldsymbol{\phi} = (\mathbf{A}^{\mathrm{T}}\mathbf{A})^{-1}\mathbf{A}^{\mathrm{T}}\Delta\boldsymbol{y}\,. \tag{6.8}$$

This process is iterated with a new estimate $\boldsymbol{\phi}^{k+1} = \boldsymbol{\phi}^{k} + \Delta\boldsymbol{\phi}$ until the error $\Delta\boldsymbol{\phi}$ becomes sufficiently small. The Gauss–Newton method has quadratic convergence, which is very fast, provided that there is a good initial estimate $\boldsymbol{\phi}^{0}$ of the parameters and the nonlinearity is not too severe. An example of a nonlinear model is the kinematic model containing the DH parameters. The nonlinearity is mild because it is due to sines and cosines, so the Gauss–Newton method usually has good convergence properties.

Both for linear and nonlinear estimation, however, rank deficiencies and numerical ill-conditioning may cause problems in inverting $\mathbf{A}^{\mathrm{T}}\mathbf{A}$. Rank deficiencies may result from two problems:

1. Inadequate data. The quality of the data for estimating the parameters can be quantified by an observability index, such as the condition number [6.4] of the regressor matrix $\mathbf{A}$. A different choice of data to maximize the observability index might result in more robust estimates. Examples are choosing different poses for kinematic calibration, or different trajectories for inertial parameter estimation.
2. Unidentifiable parameters. Perhaps no set of data from an experiment can identify some of the parameters. A procedure has to be found to eliminate or circumvent the unidentifiable parameters. Parameter elimination is usually done by using the singular value decomposition of $\mathbf{A}$, while circumvention can be achieved by using a priori values and ridge regression. This does not mean that the parameters are intrinsically identifiable, only that the experimental setup precludes their determination; for example, only one of the 10 inertial parameters of the first link of a robot manipulator can be identified if the base of the manipulator is stationary. A different experimental setup involving accelerating the base of the manipulator and measuring the reaction forces and torques at the base would allow other parameters to be identified [6.5].

Ill-conditioning may result from poor scaling of measurements or parameters:

1. The least-squares estimation minimizes the error between the predicted and measured output vector. Components y_j^l of the output vector $\boldsymbol{y}^l$ may have different units and magnitudes, such as radians versus meters, in pose measurements for kinematic calibration. Moreover, not all components may be measured to the same level of accuracy. Normalizing the output vectors with an appropriately chosen weighting matrix may result in a better conditioned estimate.
2. The parameters may also have different units and different magnitudes. This can cause problems both in terms of convergence criteria and in terms of deciding which parameters to eliminate. Again, a weighting matrix for the parameters may be introduced to improve the conditioning.

These numerical issues are generic to any parameter estimation problem, and are discussed at the end of this chapter. The next sections discuss individual robot models and issues in putting them into parameter estimation form.

6.2 Kinematic Calibration

In general, the relative location between coordinate systems requires six geometric parameters to be calibrated (position plus orientation). If there are mechanical constraints between the relative movement of the coordinate systems, such as connection by a joint, fewer parameters are required. For a rotary joint connecting two links, whose axis is a line-bound vector, four geometric parameters are required. For a prismatic joint, whose axis is a free vector, only two geometric parameters describing orientation are required.

In addition, nongeometric parameters are required to model sensors and mechanical deflection:

- Joint angle sensors may require a gain and offset determination.
- Camera calibration using an undistorted pinhole camera model may require the determination of the focal length and an image sensor offset.
- Joint flexibility from gears leads to angle change due to gravity loading and the manipulator's own weight.
- Base flexibility results from nonrigid attachment of the robot to the environment. Depending on how the manipulator is outstretched, there will be a varying effect on endpoint location.

Avidbots Ex1016
Page 161 of 2154

- Thermal effects and vibration may need to be modeled for very fine positioning control.

This section focuses on determining the geometric parameters and the sensor-based nongeometric parameters.

6.2.1 Serial-Link Robot Manipulators

The modified Denavit–Hartenberg (DH) parameters (Fig. 6.1) serve as the main geometric parameters (Sect. 2.4); the link transformation matrix is

$$^{i-1}\mathbf{T}_i = \mathrm{Rot}(\boldsymbol{x}, \alpha_i)\mathrm{Trans}(\boldsymbol{x}, a_i)\mathrm{Trans}(\boldsymbol{z}, d_i)\mathrm{Rot}(\boldsymbol{z}, \theta_i) \; . \tag{6.9}$$

- For an n-joint manipulator with rotary joint $i = 1, \ldots, n$, whose axis $\boldsymbol{z}_i$ is a line in space, all four parameters a_i, d_i, α_i, and θ_i have to be calibrated.
- For a prismatic joint i, whose axis $\boldsymbol{z}_i$ is a free vector, only two parameters describing its orientation (α_i and θ_i) are required. In principle, the axis $\boldsymbol{z}_i$ can be positioned anywhere in space. This means two DH parameters are arbitrary. One possibility is to intersect $\boldsymbol{z}_i$ with O_{i+1} [6.6, 7], which sets $d_{i+1} = 0$ and $a_{i+1} = 0$. While kinematically correct, this placement is nonintuitive in that it doesn't correspond to the physical location of the prismatic mechanism. It is possible to set a_{i+1} with a_i or θ_i to values that position $\boldsymbol{z}_i$ in the middle of the prismatic joint's mechanical structure.

In the case of nearly parallel neighboring axes, the common normal is poorly defined and the calibration is ill-conditioned. For this case, *Hayati* and *Mirmirani* [6.7] introduced an extra rotational parameter β_i about the $\boldsymbol{y}_{i-1}$ axis (Fig. 6.2).

Let $\boldsymbol{x}'_{i-1}$ lie along a line from O_i to axis $\boldsymbol{z}_{i-1}$, such that $\boldsymbol{x}'_{i-1}$ is perpendicular to $\boldsymbol{z}_i$; the intersection point defines the origin O_{i-1}. Two rotations are required to relate $\boldsymbol{z}_{i-1}$ to $\boldsymbol{z}_i = \boldsymbol{z}'_{i-1}$: a rotation α_i about $\boldsymbol{x}'_{i-1}$ maps $\boldsymbol{z}''_{i-1}$ to $\boldsymbol{z}_i$, and a rotation β_i about $\boldsymbol{y}_{i-1} = \boldsymbol{y}''_{i-1}$ maps $\boldsymbol{z}_{i-1}$ to $\boldsymbol{z}''_{i-1}$. The angle θ_i is now from $\boldsymbol{x}'_{i-1}$ to $\boldsymbol{x}_i$ about $\boldsymbol{z}_i$. The link transformation is

$$^{i-1}\mathbf{T}_i = \mathrm{Rot}(\boldsymbol{y}, \beta_i)\mathrm{Rot}(\boldsymbol{x}, \alpha_i)\mathrm{Trans}(\boldsymbol{x}, a_i)\mathrm{Rot}(\boldsymbol{z}, \theta_i) \; . \tag{6.10}$$

- For a rotary joint, the parameter β_i is calibrated instead of d_i.
- For a prismatic joint, the joint variable d_i has to be retained. As before, position $\boldsymbol{z}_i$ at some convenient location on link i, by specifying two parameters relative to the coordinate system $i+1$. Then proceed with the construction of the Hayati parameters for $d_i = 0$. As d_i changes, axis $\boldsymbol{x}_i$ is displaced along $\boldsymbol{z}_i$ relative to $\boldsymbol{x}'_{i-1}$ (not shown). The link transformation is

$$^{i-1}\mathbf{T}_i = \mathrm{Rot}(\boldsymbol{y}, \beta_i)\mathrm{Rot}(\boldsymbol{x}, \alpha_i)\mathrm{Trans}(\boldsymbol{x}, a_i)\,\mathrm{Trans}(\boldsymbol{z}, d_i)\mathrm{Rot}(\boldsymbol{z}, \theta_i) \; . \tag{6.11}$$

Although there are five parameters in this transformation, the process of setting the Hayati parameters for $d_i = 0$ is tantamount to setting the value of d_{i-1} to locate O_{i-1}, so there is no net increase in the number of parameters.

The procedures above set the coordinate systems in the intermediate links of a serial manipulator. Coordinate systems also have to be set in the base and end links, but the procedure for doing so depends on the external metrology system and on any physical constraints on the end link pose. The last frame may be n or $n+1$, while the first frame may be 0 or -1 (to keep consecutive numbers); examples appear below. Collect the unknown kinematic parameters into the vectors $\boldsymbol{a}$, $\boldsymbol{d}$, $\boldsymbol{\alpha}$, $\boldsymbol{\theta}$, and $\boldsymbol{\beta}$, and thence into the parameter vector $\boldsymbol{\phi} = \{\boldsymbol{a}, \boldsymbol{d}, \boldsymbol{\alpha}, \boldsymbol{\theta}, \boldsymbol{\beta}\}$. The parameters $\boldsymbol{\phi}$ predict the

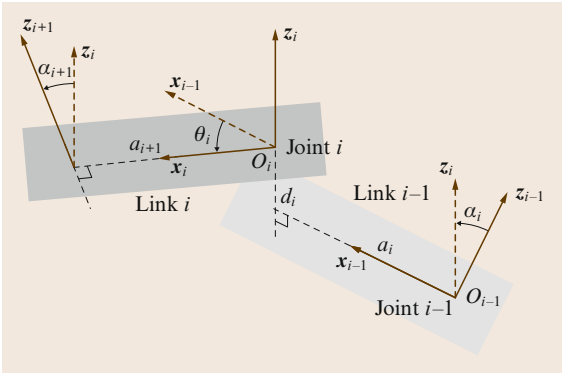


Fig. 6.1 The modified DH parameters

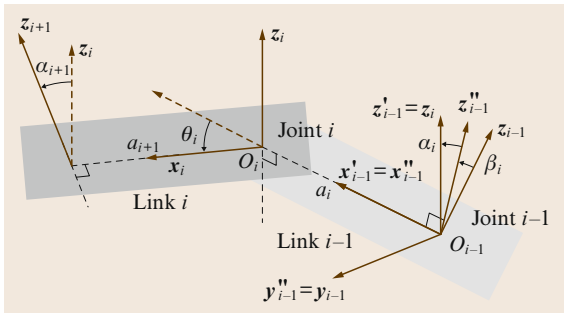


Fig. 6.2 The additional parameter β_i about $\boldsymbol{y}_{i-1}$ is employed for nearly parallel axes

Avidbots Ex1016
Page 162 of 2154

position and orientation of the last coordinate system relative to the first, such as ${}^0\mathbf{T}_{n,c}$.

Not all six components of pose are necessarily used for calibration; the number can vary from one to six. Calibration proceeds by observing the error in the prediction of a certain number of pose components, then employing the nonlinear calibration method (6.7). There are two general methods for observing error:

- *Open-loop calibration* utilizes an external metrology system to measure the pose components. Because the manipulator is not in contact with the environment during this process, the method is termed open loop.
- *Closed-loop calibration* utilizes physical constraints on the end link pose to substitute for measurements. Deviations from the physical constraint represent the prediction error. Because of the contact with the physical constraint, the manipulator forms a closed loop with the ground.

Open-Loop Kinematic Calibration

The calibration literature contains a variety of metrology systems, which can be categorized based on the number of pose components measured [6.8]:

- *1 component*: The distance to a single point on the end link can be measured in a variety of ways, such as an instrumented ball bar [6.9], wire potentiometer [6.10], or laser displacement meter [6.11].
- *2 components*: A single theodolite has been employed to provide two orientation measurements [6.12]. A reference length had to be sighted for scaling.
- *3 components*: Laser tracking systems provide accurate three-dimensional (3-D) measurements by reflecting a beam off a retroreflector mounted on the end effector. The beam provides length information, while the gimbal drive for laser steering provides two orientation measurements [6.13]. Since the least precise part of this setup is the angle sensing, another approach is to employ three laser tracking systems and use only the length information. Commercial 3-D stereo-camera motion tracking systems also provide high-accuracy measurements of position.
- *5 components*: *Lau* et al. [6.14] presented a steerable laser interferometer with steerable reflector. With pitch and yaw measurements, the steerable interferometer yields all three components of position, while the steerable reflector yields two components of orientation.
- *6 components*: Full pose can be inferred from the 3-D position of multiple points on the last link measured with a stereo-camera system. A coordinate system fit to the cloud of points yields position plus orientation [6.15].
  Vincze et al. [6.16] measured full pose with a single-beam laser tracking system, by fitting the robot with a retroreflector mounted on a universal joint. Position is measured using interferometry, as usual. The novel aspect is orientation measurement, by imaging of the diffraction pattern of the edges of the retroreflector.

Examples are given for calibrating with three pose components and with all six pose components measured.

Point Measurement. The 3-D position of a particular point on the end link can be conveniently located by some form of stereo camera system. The camera system defines a global coordinate system relative to which frame 1 of the robot is located. To provide enough parameters, an intermediate coordinate system has to be introduced; this intermediate coordinate system has index 0, while the camera system is given index -1 to keep consecutive numbers. Two of the eight parameters are arbitrary. Figure 6.3 shows one possibility: z_0 is made parallel ($\alpha_0 = 0$) and coincident ($a_0 = 0$) with z_{-1}. The calibrated parameters are d_0, θ_0, a_1, d_1, α_1, and θ_1. In the case that z_{-1} is nearly parallel to z_0, the measurement coordinate system can be simply redefined to avoid using Hayati parameters; for example, y_{-1} can be redefined as z_{-1}.

In the end link, the origin O_n and axis x_n are unspecified, as are the associated parameters d_n and θ_n. Locating the measured point requires only three parameters, so the addition of a single coordinate system $n+1$ is required to provide an additional parameter.

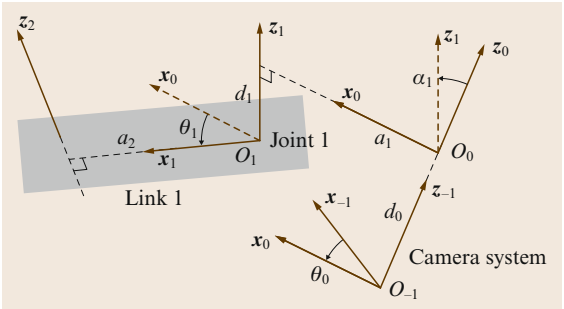


Fig. 6.3 A camera system (index−1) is located relative to the manipulator coordinate system 1 via an intermediate coordinate system 0

The measured point is defined as the origin O_{n+1} of the new coordinate system, and the normal from z_n that intersects O_{n+1} defines the x_n axis (Fig. 6.4). Three parameters are arbitrary; a simple choice is to make z_{n+1} parallel to z_n ($\alpha_{n+1} = 0$) and x_{n+1} collinear with x_n ($\theta_{n+1} = 0$ and $d_{n+1} = 0$). The calibrated parameters are a_{n+1}, d_n, and θ_n.

From the transformation

$$^{-1}\mathbf{T}_{n+1} = {}^{-1}\mathbf{T}_0 \dots {}^{n}\mathbf{T}_{n+1} ,$$

the position $^{-1}\boldsymbol{p}_{n+1}$ of the measured point relative to the camera frame is extracted. Collect the unknown kinematic parameters into vectors $\boldsymbol{a}, \boldsymbol{d}, \boldsymbol{\alpha}, \boldsymbol{\theta}$, and $\boldsymbol{\beta}$, and thence into the parameter vector $\boldsymbol{\phi} = \{\boldsymbol{a}, \boldsymbol{d}, \boldsymbol{\alpha}, \boldsymbol{\theta}, \boldsymbol{\beta}\}$. The nonlinear kinematic model analogous to (6.2) is

$$^{-1}\boldsymbol{p}^l_{n+1} = \boldsymbol{f}(\boldsymbol{q}^l, \boldsymbol{\phi}) , \quad l = 1, \dots, P , \tag{6.12}$$

where $\boldsymbol{q}^l$ is the vector of joint variables for pose l. To linearize this equation into estimation form (6.6), the associated Jacobians for each parameter type are calculated

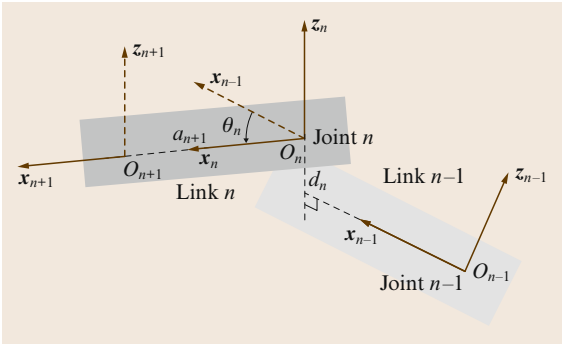


Fig. 6.4 A measured point O_{n+1} is located on the end link by adding a coordinate system $n+1$

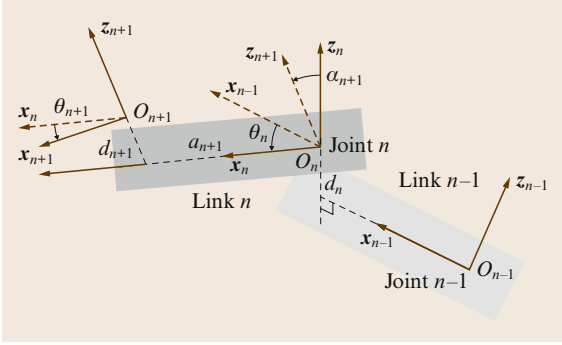


Fig. 6.5 Full pose measurement of coordinate system $n+1$ in the end link

$$\Delta^{-1}\boldsymbol{p}^l_{n+1} = {}^{-1}\boldsymbol{p}^l_{n+1} - {}^{-1}\boldsymbol{p}^l_{n+1,c} = \mathbf{J}^l \Delta\boldsymbol{\phi}$$
$$= \begin{pmatrix} \mathbf{J}^l_{\boldsymbol{a}} & \mathbf{J}^l_{\boldsymbol{d}} & \mathbf{J}^l_{\boldsymbol{\alpha}} & \mathbf{J}^l_{\boldsymbol{\theta}} & \mathbf{J}^l_{\boldsymbol{\beta}} \end{pmatrix} \begin{pmatrix} \Delta\boldsymbol{a} \\ \Delta\boldsymbol{d} \\ \Delta\boldsymbol{\alpha} \\ \Delta\boldsymbol{\theta} \\ \Delta\boldsymbol{\beta} \end{pmatrix} , \tag{6.13}$$

where a typical column i for each individual parameter Jacobian is derived from the screw parameters as if each parameter represented an active joint (Sect. 2.8.1)

$$\mathbf{J}^l_{a_i} = \begin{cases} {}^{-1}\boldsymbol{x}^l_{i-1} & \text{DH} \\ {}^{-1}\boldsymbol{x}'^l_{i-1} & \text{Hayati} \end{cases} , \tag{6.14}$$

$$\mathbf{J}^l_{d_i} = {}^{-1}\boldsymbol{z}^l_i , \tag{6.15}$$

$$\mathbf{J}^l_{\alpha_i} = \begin{cases} {}^{-1}\boldsymbol{x}^l_{i-1} \times {}^{-1}\boldsymbol{d}^l_{i-1,n+1} & \text{DH} \\ {}^{-1}\boldsymbol{x}'^l_{i-1} \times {}^{-1}\boldsymbol{d}^l_{i-1,n+1} & \text{Hayati} \end{cases} , \tag{6.16}$$

$$\mathbf{J}^l_{\theta_i} = {}^{-1}\boldsymbol{z}^l_i \times {}^{-1}\boldsymbol{d}^l_{i,n+1} , \tag{6.17}$$

$$\mathbf{J}^l_{\beta_i} = {}^{-1}\boldsymbol{y}^l_{i-1} \times {}^{-1}\boldsymbol{d}^l_{i-1,n+1} , \tag{6.18}$$

where $^{-1}\boldsymbol{d}_{i,n+1} = {}^{-1}\mathbf{R}_i {}^{i}\boldsymbol{p}_{n+1}$ refers the interorigin vector to coordinate system -1. Equation (6.13) is stacked for all the poses for the final estimation form

$$\Delta^{-1}\boldsymbol{p}_{n+1} = \mathbf{J}\Delta\boldsymbol{\phi} , \tag{6.19}$$

which corresponds to (6.7) and which is solved for $\Delta\boldsymbol{\phi}$ by least squares and for $\boldsymbol{\phi}$ by iteration.

Full Pose Measurement. Suppose that coordinate system $n+1$ in link n is measured (Fig. 6.5). Coordinate system n is completed in the usual way, and the six calibrated parameters to locate coordinate system $n+1$ are d_n, α_n, θ_n, a_{n+1}, d_{n+1}, and θ_{n+1}. If z_{n+1} is nearly parallel to z_n, this axis may simply be permuted with other axes, such as y_{n+1}, to avoid using the Hayati parameters.

In addition to the position (6.12), orientation equations of coordinate system $n+1$

$$^{-1}\mathbf{R}^l_{n+1} = \mathbf{F}(\boldsymbol{q}^l, \boldsymbol{\phi}) , \quad l = 1, \dots, P \tag{6.20}$$

are extracted from $^{-1}\mathbf{T}_{n+1}$, where $\mathbf{F}$ is a matrix function. Linearization of this equation yields

$$\Delta^{-1}\mathbf{R}^l_{n+1} = {}^{-1}\mathbf{R}^l_{n+1} - {}^{-1}\mathbf{R}^l_{n+1,c}$$
$$= \Delta^{-1}\boldsymbol{\rho}^l_{n+1} \times {}^{-1}\mathbf{R}^l_{n+1,c}$$
$$\mathbf{S}(\Delta^{-1}\boldsymbol{\rho}^l_{n+1}) = \left({}^{-1}\mathbf{R}^l_{n+1} - {}^{-1}\mathbf{R}^l_{n+1,c}\right)\left({}^{-1}\mathbf{R}^l_{n+1,c}\right)^{\mathrm{T}} , \tag{6.21}$$

Avidbots Ex1016
Page 164 of 2154

where $\Delta^{-1}\boldsymbol{\rho}^l_{n+1}$ are differential orthogonal rotations which are the finite-difference counterpart to the angular velocity vector. Continuing with this analogy, a Jacobian $\mathbf{J}$ similar to that for spatial velocities (Sect. 2.8.1) is employed to represent the combined effect of parameter variations $\Delta\boldsymbol{\phi}$ on changes in position $\Delta^{-1}\boldsymbol{p}^l_{n+1}$ and orientation $\Delta^{-1}\boldsymbol{\rho}^l_{n+1}$

$$\begin{pmatrix}\Delta^{-1}\boldsymbol{p}^l_{n+1}\\ \Delta^{-1}\boldsymbol{\rho}^l_{n+1}\end{pmatrix} = \mathbf{J}^l\Delta\boldsymbol{\phi}\,. \tag{6.22}$$

Compared to (6.13), the Jacobian $\mathbf{J}^l$ now has six rows, as do the individual parameter Jacobians

$$\mathbf{J}^l_{a_i} = \begin{cases}\begin{pmatrix}{}^{-1}\boldsymbol{x}^l_{i-1}\\ \mathbf{0}\end{pmatrix} & \text{DH}\\ \begin{pmatrix}{}^{-1}\boldsymbol{x'}^l_{i-1}\\ \mathbf{0}\end{pmatrix} & \text{Hayati}\end{cases}, \tag{6.23}$$

$$\mathbf{J}^l_{d_i} = \begin{pmatrix}{}^{-1}\boldsymbol{z}^l_i\\ \mathbf{0}\end{pmatrix}, \tag{6.24}$$

$$\mathbf{J}^l_{\alpha_i} = \begin{cases}\begin{pmatrix}{}^{-1}\boldsymbol{x}^l_{i-1}\times{}^{-1}\boldsymbol{d}^l_{i-1,n+1}\\ {}^{-1}\boldsymbol{x}^l_{i-1}\end{pmatrix} & \text{DH}\\ \begin{pmatrix}{}^{-1}\boldsymbol{x'}^l_{i-1}\times{}^{-1}\boldsymbol{d}^l_{i-1,n+1}\\ {}^{-1}\boldsymbol{x'}^l_{i-1}\end{pmatrix} & \text{Hayati}\end{cases}, \tag{6.25}$$

$$\mathbf{J}^l_{\theta_i} = \begin{pmatrix}{}^{-1}\boldsymbol{z}^l_i\times{}^{-1}\boldsymbol{d}^l_{i,n+1}\\ {}^{-1}\boldsymbol{z}^l_i\end{pmatrix}, \tag{6.26}$$

$$\mathbf{J}^l_{\beta_i} = \begin{pmatrix}{}^{-1}\boldsymbol{y}^l_{i-1}\times{}^{-1}\boldsymbol{d}^l_{i-1,n+1}\\ {}^{-1}\boldsymbol{y}^l_{i-1}\end{pmatrix}. \tag{6.27}$$

As before, $\Delta\boldsymbol{\phi}$ is found by least squares and $\boldsymbol{\phi}$ by iteration.

Closed-Loop Calibration

Physical constraints on end-effector position or orientation can substitute for measurements. The location of a physical constraint defines the reference coordinate system; the *measurement* of endpoint position or orientation is therefore defined as zero. The deviation from the physical constraint due to an incorrect kinematic model is cast as a displacement from the reference coordinates. Analogous to point measurement and full pose measurement, there are closed-loop methods using point constraints and full pose constraints.

Point Constraint. Suppose that the end-effector holds a stylus that makes contact with a fixed point in the environment. The orientation of the stylus can be changed by varying the joint angles, as long as the point of contact is not changed. Before, the measurement system for point measurement defined the reference coordinate system -1 (Fig. 6.3), and there was an end effector coordinate system $n+1$ whose origin O_{n+1} was measured (Fig. 6.4). Now the reference coordinate origin O_0 has index 0 and is collocated with O_{n+1} (Fig. 6.6). An extra coordinate system -1 is not required because a point is located relative to coordinate system 1 with only three parameters: a_1, d_1, and θ_1. The arbitrary choice $\alpha_1 = 0$ is made, i. e., z_0 is made parallel to z_1.

Different poses while maintaining the point contact can be generated manually, or they can be generated automatically if there is a force control capability. In comparison to (6.13), the *measured* position ${}^0\boldsymbol{p}^l_{n+1} = 0$ by definition and the linearized calibration equation is simply

$$\Delta^{-1}\boldsymbol{p}^l_{n+1} = -{}^{-1}\boldsymbol{p}^l_{n+1,c} = \mathbf{J}^l\Delta\boldsymbol{\phi}\,. \tag{6.28}$$

The set of poses that may be generated is limited relative to open-loop calibration, which may affect identifiability.

Full Pose Constraint. Analogous to full pose measurement (Fig. 6.5), the end link may be fully constrained by rigidly gripping the environment. If the manipulator is redundant (Chap. 10), then poses can be generated through self motion. Generating such poses would require endpoint force/torque sensing or joint torque sensing to be accomplished.

The end link can be considered part of the ground due to rigid attachment, and therefore one fewer coordinate system is required than for the fixed point case. Fig. 6.7 shows one way of setting up coordinate systems 0 and n for calibration. Axis z_0 is set equal and coincident with axis z_n. The common normal between z_0 and z_1 sets the origin O_0 and the axis x_0. Coordinate system n is completed by defining $O_n = O_0$ and $x_n = x_0$. Six parameters result for calibration, as they must to re-

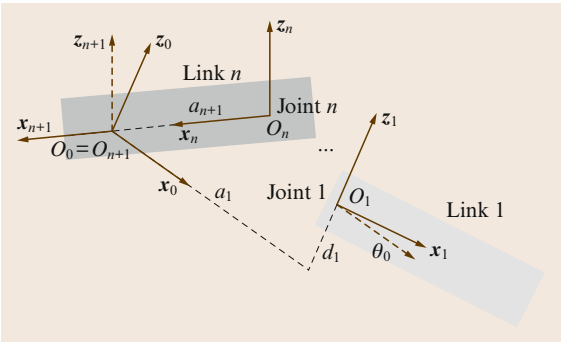


Fig. 6.6 Fixed point contact, setting $O_0 = O_{n+1}$

late coordinate system n to coordinate system 0: θ_n, d_n, α_1, a_1, d_1, and θ_1.

Similar to (6.28) after index adjustment, the *measured* position ${}^0\boldsymbol{p}_n^l = 0$ by definition and the linearized position calibration equation is

$$\Delta {}^0\boldsymbol{p}_n^l = -{}^0\boldsymbol{p}_{n,c}^l \,. \tag{6.29}$$

With regard to the error in orientation (6.21), the *measured* orientation ${}^0\mathbf{R}_n^l = \mathbf{I}$, the identity matrix, after index adjustment.

$$\mathbf{S}\left(\Delta {}^0\boldsymbol{\rho}_n^l\right) = \left(\mathbf{I} - {}^0\mathbf{R}_{n,c}^l\right)\left({}^0\mathbf{R}_{n,c}^l\right)^{\mathrm{T}} = \left({}^0\mathbf{R}_{n,c}^l\right)^{\mathrm{T}} - \mathbf{I} \,. \tag{6.30}$$

The error (6.22) is then applied as for full pose measurement, after index adjustment. The set of poses that may be generated is limited relative to open-loop calibration, which may affect identifiability.

6.2.2 Parallel Manipulator Calibration

Parallel manipulators are comprised of multiple closed loops. The methods of the previous section could readily be extended to calibrate parallel manipulators. Rather than treat parallel and serial manipulators differently, and continue to elaborate the calibration equations on a case-by-case basis that includes different loop arrangements and different sensing arrangements, *Hollerbach* and *Wampler* [6.8] presented a unifying methodology termed the *calibration index*, which views all calibration problems as closed-loop calibration problems.

Open-loop calibration is converted to closed-loop calibration by considering the end effector measurement to form a joint. All measurements, whether from joints or metrology systems, are put on an equal footing, as are all unsensed joints, whether from unsensed components of pose, passive environmental constraints, or joints in the chain without sensors. In the case of parallel linkages, sufficient numbers of loop-closure equations are formulated to characterize the kinematics and are combined at each pose. Because the loop-closure equations are implicit functions of all measurements, *Wampler* et al. [6.2] called this calibration method the *implicit loop method*.

Fig. 6.8 illustrates this idea for a calibrated stereo camera metrology system that measures the 3-D coordinates of a distinguished point on the end-effector of an uncalibrated robot. On the right this camera system has been replaced by a prismatic leg, which stylistically represents a six-degree-of-freedom (6-DOF) joint that provides equivalent 3-D coordinate measurements. The result is a closed-loop mechanism.

The kinematic loop closure equations $\boldsymbol{f}$ for the i-th pose ($i = 1, \ldots, P$) are formed as

$$\boldsymbol{f}^i(\boldsymbol{\phi}) \equiv \boldsymbol{f}(\boldsymbol{x}^i, \boldsymbol{\phi}) = \mathbf{0} \,, \tag{6.31}$$

where $\boldsymbol{\phi}$ is a vector of robot parameters to be calibrated, $\boldsymbol{x}^i$ is a vector of joint sensor readings and possibly external sensor readings, and $\boldsymbol{f}^i$ absorbs the sensor readings $\boldsymbol{x}^i$ and so is given an index. Combining (6.31) for the P poses into a single matrix equation

$$\boldsymbol{f}(\boldsymbol{\phi}) = \left(\boldsymbol{f}^{1\mathrm{T}} \quad \ldots \quad \boldsymbol{f}^{P\mathrm{T}}\right)^{\mathrm{T}} = \mathbf{0} \,. \tag{6.32}$$

Linearize (6.32) around the nominal values of the parameters

$$\Delta \boldsymbol{f} = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{\phi}} \Delta \boldsymbol{\phi} = \mathbf{A} \Delta \boldsymbol{\phi} \,, \tag{6.33}$$

where $\Delta \boldsymbol{f}$ is the deviation of the computed loop closure equations from zero, $\mathbf{J}$ is the identification Jacobian, and $\Delta \boldsymbol{\phi}$ is the correction to be applied to the current parameter estimate. The calibration problem is then solved by minimizing $\Delta \boldsymbol{f}$ via iterative least squares.

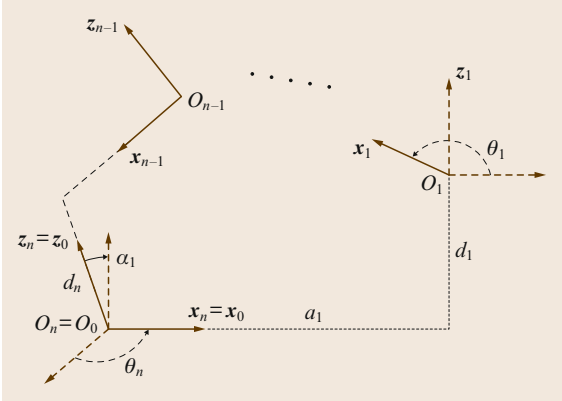


Fig. 6.7 Fully constrained end link, setting $O_0 = O_n$

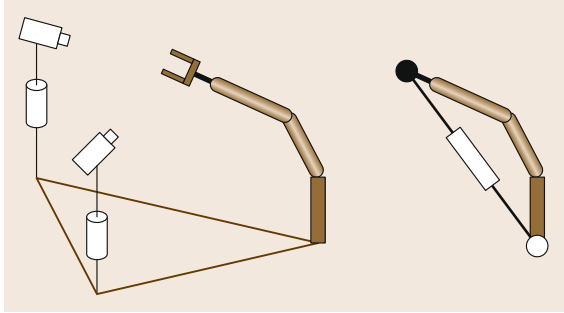


Fig. 6.8 External metrology system modeled as a 6-DOF joint

Avidbots Ex1016
Page 166 of 2154

The Calibration Index

The basis for kinematic calibration is to compute the error in positioning a manipulator using the current kinematic model. The error may be relative to measurements by an external metrology system (open-loop methods), or relative to a physical constraint such as contact with a plane (closed-loop methods). An external metrology system may measure all six components of pose or fewer components such as a point on the end link (three components). Likewise, a physical constraint can limit between one and six components of pose. Constraints and measurements may be mixed. The number of measurements and constraints determine how many scalar equations per pose are available for calibration.

The calibration index C quantifies the number of equations per pose. While this analysis is rather straightforward for serial linkages, for parallel linkages it can be difficult to infer the number of independent equations per pose

$$C = S - M \,, \tag{6.34}$$

where S is the sensor index and M is the mobility index. The mobility index [6.17] characterizes the degrees of freedom in the calibration setup.

$$M = 6n - \sum_{i=1}^{N_J} n_i^{\mathrm{c}} \,, \tag{6.35}$$

where n is the number of links, N_J is the number of joints, and n_i^{c} is the number of constraints at joint i. n includes any extra links attached to the robot to constrain or measure its motion. N_J includes joints of any additional linkages added for calibration. For a rotational or prismatic joint $n_i^{\mathrm{c}} = 5$, while for a ball or spherical joint $n_i^{\mathrm{c}} = 3$. For an external measurement system for a freely moving end-effector $n_{N_J}^{\mathrm{c}} = 0$, while for rigid attachment of the endpoint $n_{N_J}^{\mathrm{c}} = 6$. While generally correct, there are exceptions to (6.35) due to special or degenerate mechanisms that have to be analyzed on a case-by-case basis.

The sensor index S is the total number of sensors in the joints

$$S = \sum_{i=1}^{N_J} S_i \,, \tag{6.36}$$

where S_i is the number of sensed degrees in joint i. Usually $S_i = 1$ for the lower-order pairs typical of actuated joints, while $S_{N_J} = 6$ for full pose measurement of the end-effector *joint* N_J. For an unsensed joint, such as in passive environment kinematics, $S_i = 0$.

If P is the number of poses, then CP is the total number of equations for the calibration procedure. Clearly a larger C means that fewer poses are required, other things being equal. For the single-loop case consisting of a series of sensed lower-order robot joints ($S_i = 1$, $n_i^{\mathrm{c}} = 5$, $i = 1, \ldots, N_J - 1$) with a final joint $(S_{N_J}, n_{N_J}^{\mathrm{c}})$ connecting the end-effector to ground, one has from (6.34–6.35)

$$C = S_{N_J} + n_{N_J}^{\mathrm{c}} \,. \tag{6.37}$$

According to the calibration index, using full endpoint constraints is an equivalent kinematic calibration method to using full pose measurements. There is potentially a problem with the smaller range of poses available with the endpoint constrained, but otherwise the mathematics are the same.

Categorization of Serial Link Calibration Methods

A great variety of calibration methods have been proposed, varying in the manner of pose measurement or endpoint constraints. These are categorized below according to the calibration index C and the values of $n_{N_J}^{\mathrm{c}}$ and S_{N_J}:

- $C = 6$: $n_{N_J}^{\mathrm{c}} = 0$ and $S_{N_J} = 6$ corresponds to full pose measurement. $n_{N_J}^{\mathrm{c}} = 6$ and $S_{N_J} = 0$ corresponds to rigid endpoint attachment.
- $C = 5$: $n_{N_J}^{\mathrm{c}} = 0$ and $S_{N_J} = 5$ corresponds to 5-DOF pose measurement [6.14]. $n_{N_J}^{\mathrm{c}} = 5$ and $S_{N_J} = 0$ corresponds to 5-DOF endpoint constraints, such as manipulation of an unsensed passive hinge joint [6.18].
- $C = 4$: No published method exists for $C = 4$.
- $C = 3$: $n_{N_J}^{\mathrm{c}} = 0$ and $S_{N_J} = 3$ corresponds to 3-DOF pose measurement. $n_{N_J}^{\mathrm{c}} = 3$ and $S_{N_J} = 0$ corresponds to 3-DOF endpoint constraints.
- $C = 2$: $n_{N_J}^{\mathrm{c}} = 0$ and $S_{N_J} = 2$ corresponds to 2-DOF pose measurements, such as is provided by a single theodolite [6.12]. $n_{N_J}^{\mathrm{c}} = 2$ and $S_{N_J} = 0$ corresponds to 2-DOF endpoint constraints. Motion along a line provides two constraints [6.19].
- $C = 1$: $n_{N_J}^{\mathrm{c}} = 0$ and $S_{N_J} = 1$ corresponds to measurement of just 1-DOF pose, provided by a linear transducer such as an linear variable differential transformer (LVDT) [6.9] or wire potentiometer [6.10]. $n_{N_J}^{\mathrm{c}} = 1$ and $S_{N_J} = 0$ corresponds to a plane constraint [6.20].

Categorization of Parallel-Link Calibration Methods

To calibrate parallel robots, a loop-closure equation is written for each loop j

$$\mathbf{0} = \boldsymbol{f}_j(\boldsymbol{\phi}) \,. \tag{6.38}$$

Avidbots Ex1016
Page 167 of 2154

The equations for all the loops are combined. One problem is to eliminate unsensed degrees of freedom. The calibration index is now applied based on the number of loops:

- *2 loops*: A two-loop mechanism has three arms or legs, attached to a common platform. An example is the RSI Research Ltd. Hand Controller [6.21], which employs three 6-DOF arms with three sensed joints each. The mobility of this mechanism is $M = 6$. As $S = 9$, therefore $C = 3$ and closed-loop calibration is possible.
- *4 loops*: *Nahvi* et al. [6.22] calibrated a spherical shoulder joint, driven redundantly by four prismatic legs (Fig. 6.9). In addition, the platform is constrained to rotate about a spherical joint; with the four legs, four kinematic loops are formed. For this system, $M = 3$ and $S = 4$, so that $C = 1$. Hence self-calibration is possible. Without the extra leg and the sensing it provides, one would have $C = 0$ and calibration would not be possible.
- *5 loops*: *Wampler* et al. [6.2] calibrated the six-legged Stewart platform ($M = 6$) via a closed-loop procedure. In addition to leg length measurements, all angles on one of the legs were measured ($S = 11$). The reason for the extra sensors is to yield a unique forward kinematics solution, but a side benefit is that, with $C = 5$, closed-loop calibration is possible. For a regular Stewart platform without the instrumented leg, $S = 6$ and hence $C = 0$. External pose measurement is required; for example, with full pose measurement $S = 12$ and $C = 6$; full pose measurement was utilized in [6.23].

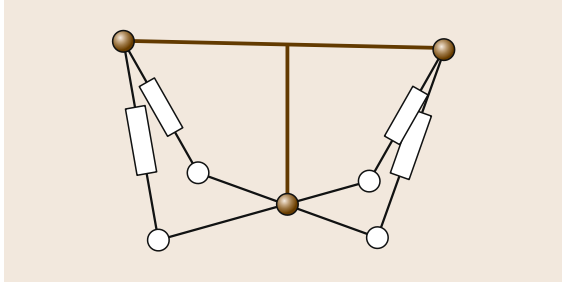


Fig. 6.9 Redundant parallel drive shoulder joint

6.3 Inertial Parameter Estimation

A rigid body i has 10 inertial parameters: mass m_i, center of mass $\boldsymbol{r}_{0i}$ relative to an origin O_0, and symmetric inertia matrix $\bar{\mathbf{I}}_i$ referred here to the origin O_i (Fig. 6.10). The rigid body may be a load held by the end effector, or it can be one of the manipulator's own links. By generating a trajectory and measuring forces or torques in combination with velocity and acceleration, some or all of the inertial parameters can be estimated.

6.3.1 Link Inertial Parameter Estimation

The procedure involves:

1. Formulating the Newton–Euler equations of the load dynamics to reveal a linear dependence on the inertial parameters.
2. Estimating the parameters using ordinary least squares.

Begin with the Newton-Euler equations of the last link, and assume that suitable filters have been designed to estimate the velocity $\dot{\theta}_i$ and acceleration $\ddot{\theta}_i$ at each joint i.

The center of mass of link n is defined as C_n, located relative to the base origin O_0 by $\boldsymbol{r}_n = C_n - O_0$ and relative to link n's origin O_n by $\boldsymbol{c}_n = C_n - O_n$. All vectors and matrices are expressed in coordinate system n, in which the center of mass location $\boldsymbol{c}_n$ and the inertia matrix $\bar{\mathbf{I}}_n$ about origin O_n are constant.

From (3.35) in Chap. 3, the Newton-Euler equations referred to O_n are

$$\boldsymbol{f}_n = \mathbf{I}_n \boldsymbol{a}_n + \boldsymbol{v}_n \times \mathbf{I}_n \boldsymbol{v}_n \,. \tag{6.39}$$

Substituting for the spatial inertia $\mathbf{I}_n$, the spatial acceleration $\boldsymbol{a}_n$, and the spatial velocity $\boldsymbol{v}_n$, the first term on

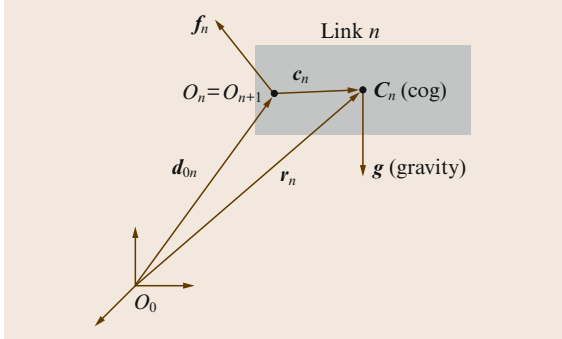


Fig. 6.10 Location of the center of gravity for an intermediate link i, and the constraint forces and torques

the right evaluates to

$$\mathbf{I}_n \boldsymbol{a}_n = \begin{pmatrix} \bar{\mathbf{I}}_n & m_n \mathbf{S}(\boldsymbol{c}_n) \\ m_n \mathbf{S}(\boldsymbol{c}_n)^{\mathrm{T}} & m_n \mathbf{1} \end{pmatrix} \begin{pmatrix} \dot{\boldsymbol{\omega}}_n \\ \dot{\boldsymbol{v}}_n \end{pmatrix} = \begin{pmatrix} \bar{\mathbf{I}}_n \dot{\boldsymbol{\omega}}_n + m_n \mathbf{S}(\boldsymbol{c}_n) \left(\ddot{\boldsymbol{d}}_{0n} - \boldsymbol{\omega}_n \times \boldsymbol{v}_n \right) \\ m_n \mathbf{S}(\boldsymbol{c}_n)^{\mathrm{T}} \dot{\boldsymbol{\omega}}_n + m_n \left(\ddot{\boldsymbol{d}}_{0n} - \boldsymbol{\omega}_n \times \boldsymbol{v}_n \right) \end{pmatrix},$$

where $\dot{\boldsymbol{v}}_n = \ddot{\boldsymbol{d}}_{0n} - \boldsymbol{\omega}_n \times \boldsymbol{v}_n$ has been substituted. The second term on the right evaluates to

$$\boldsymbol{v}_n \times \mathbf{I}_n \boldsymbol{v}_n = \begin{pmatrix} \mathbf{S}(\boldsymbol{\omega}_n) & \mathbf{S}(\boldsymbol{v}_n) \\ \mathbf{0} & \mathbf{S}(\boldsymbol{\omega}_n) \end{pmatrix} \times \begin{pmatrix} \bar{\mathbf{I}}_n & m_n \mathbf{S}(\boldsymbol{c}_n) \\ m_n \mathbf{S}(\boldsymbol{c}_n)^{\mathrm{T}} & m_n \mathbf{1} \end{pmatrix} \begin{pmatrix} \boldsymbol{\omega}_n \\ \boldsymbol{v}_n \end{pmatrix} = \begin{pmatrix} \mathbf{S}(\boldsymbol{\omega}_n) \bar{\mathbf{I}}_n \boldsymbol{\omega}_n + m_n \mathbf{S}(\boldsymbol{c}_n) \mathbf{S}(\boldsymbol{\omega}_n) \boldsymbol{v}_n \\ \mathbf{S}(\boldsymbol{\omega}_n) m_n \mathbf{S}(\boldsymbol{c}_n)^{\mathrm{T}} \boldsymbol{\omega}_n + \mathbf{S}(\boldsymbol{\omega}_n) m_n \boldsymbol{v}_n \end{pmatrix}.$$

Combining and simplifying,

$$\boldsymbol{f}_n = \begin{pmatrix} \bar{\mathbf{I}}_n \dot{\boldsymbol{\omega}}_n + \mathbf{S}(\boldsymbol{\omega}_n) \bar{\mathbf{I}}_n \boldsymbol{\omega}_n - \mathbf{S}\left(\ddot{\boldsymbol{d}}_{0n} \right) m_n \boldsymbol{c}_n \\ m_n \ddot{\boldsymbol{d}}_{0n} + \mathbf{S}(\dot{\boldsymbol{\omega}}_n) m_n \boldsymbol{c}_n + \mathbf{S}(\boldsymbol{\omega}_n) \mathbf{S}(\boldsymbol{\omega}_n) m_n \boldsymbol{c}_n \end{pmatrix}, \tag{6.40}$$

where the mass moment $m_n \boldsymbol{c}_n$ appears as a quantity to be estimated in combination. However, since the mass m_n is separately estimated from the $m_n \ddot{\boldsymbol{d}}_{0n}$ term, the center of mass $\boldsymbol{c}_n$ can be extracted. To account explicitly for gravity $\boldsymbol{g}$, we substitute $\ddot{\boldsymbol{d}}_{0n} - \boldsymbol{g}$ for $\ddot{\boldsymbol{d}}_{0n}$ subsequently.

To formulate an estimation algorithm, the force and torque measured by the wrist sensor must be expressed in terms of the product of known geometric parameters and the unknown inertial parameters. Elements of the inertia matrix are vectorized as $\boldsymbol{l}(\bar{\mathbf{I}}_n)$ according to the following notation

$$\bar{\mathbf{I}}_n \boldsymbol{\omega}_n = \begin{pmatrix} \omega_1 & \omega_2 & \omega_3 & 0 & 0 & 0 \\ 0 & \omega_1 & 0 & \omega_2 & \omega_3 & 0 \\ 0 & 0 & \omega_1 & 0 & \omega_2 & \omega_3 \end{pmatrix} \begin{pmatrix} I_{11} \\ I_{12} \\ I_{13} \\ I_{22} \\ I_{23} \\ I_{33} \end{pmatrix} \equiv \mathbf{L}(\boldsymbol{\omega}_n) \boldsymbol{l}\left(\bar{\mathbf{I}}_n \right),$$

where $\mathbf{L}(\boldsymbol{\omega}_n)$ is a 3×6 matrix of angular velocity elements and

$$\bar{\mathbf{I}}_n = \begin{pmatrix} I_{11} & I_{12} & I_{13} \\ I_{12} & I_{22} & I_{23} \\ I_{13} & I_{23} & I_{33} \end{pmatrix}.$$

Using these expressions, (6.40) can be written as

$$\boldsymbol{f}_n = \begin{pmatrix} \mathbf{0} & -\mathbf{S}\left(\ddot{\boldsymbol{d}}_{0n} \right) & \mathbf{L}(\dot{\boldsymbol{\omega}}_n) + \mathbf{S}(\boldsymbol{\omega}_n) \mathbf{L}(\boldsymbol{\omega}_n) \\ \ddot{\boldsymbol{d}}_{0n} & \mathbf{S}(\dot{\boldsymbol{\omega}}_n) + \mathbf{S}(\boldsymbol{\omega}_n) \mathbf{S}(\boldsymbol{\omega}_n) & \mathbf{0} \end{pmatrix} \times \begin{pmatrix} m_n \\ m_n \boldsymbol{c}_n \\ \boldsymbol{l}\left(\bar{\mathbf{I}}_n \right) \end{pmatrix}$$

or more compactly,

$$\boldsymbol{f}_n = \mathbf{A}_n \boldsymbol{\phi}_n, \tag{6.41}$$

where $\mathbf{A}_n$ is a 6×10 matrix, and $\boldsymbol{\phi}_n$ is the vector of the 10 unknown inertial parameters which appear linearly.

The previous formulation can be extended to all the links of a manipulator with n joints (Fig. 6.10). (Only manipulators with revolute joints will be considered, since handling prismatic joints requires only trivial modifications to the algorithm.) Define $\boldsymbol{f}_{ij}$ as the spatial force at joint i due to movement of link j alone. Then $\boldsymbol{f}_{ii}$ is the spatial force at joint i due to movement of its own link, and is the same as (6.41) with i substituted for n in the $\mathbf{A}_n$ matrix

$${}^i\boldsymbol{f}_{ii} = {}^i\mathbf{A}_i \boldsymbol{\phi}_i, \tag{6.42}$$

where $\boldsymbol{\phi}_i$ is the vector of unknown link i inertial parameters. Superscript i has been added to indicate that vectors are expressed in terms of link i coordinates, so that the center of mass ${}^i\boldsymbol{c}_i$ and the inertia matrix ${}^i\bar{\mathbf{I}}_i$ are constant.

The total spatial force ${}^i\boldsymbol{f}_i$ at joint i is the sum of the spatial forces ${}^i\boldsymbol{f}_{ij}$ for all links j distal to joint i

$${}^i\boldsymbol{f}_i = \sum_{j=i}^{n} {}^i\boldsymbol{f}_{ij}. \tag{6.43}$$

Each spatial force ${}^i\boldsymbol{f}_{ij}$ at joint i is determined by transmitting the distal spatial force ${}^j\boldsymbol{f}_{jj}$ across intermediate joints. Using the spatial force transform matrix ${}^i\mathbf{X}_j^{\mathrm{F}}$,

$${}^i\boldsymbol{f}_{i,i+1} = {}^i\mathbf{X}_{i+1}^{\mathrm{F}} \, {}^{i+1}\boldsymbol{f}_{i+1,i+1} = {}^i\mathbf{X}_{i+1}^{\mathrm{F}} \, {}^{i+1}\mathbf{A}_{i+1} \boldsymbol{\phi}_{i+1}. \tag{6.44}$$

For convenience, we note that ${}^i\mathbf{X}_i^{\mathrm{F}} = \mathbf{1}_{6\times 6}$. To obtain the forces and torques at the i-th joint due to the movements of the j-th link, these matrices can be cascaded

$$\begin{aligned} {}^i\boldsymbol{f}_{ij} &= {}^i\mathbf{X}_{i+1}^{\mathrm{F}} \quad {}^{i+1}\mathbf{X}_{i+2}^{\mathrm{F}} \quad \cdots \quad {}^{j-1}\mathbf{X}_j^{\mathrm{F}} \quad {}^j\boldsymbol{f}_{jj} \\ &= {}^i\mathbf{X}_j^{\mathrm{F}j} \mathbf{A}_j \boldsymbol{\phi}_j. \end{aligned} \tag{6.45}$$

Avidbots Ex1016
Page 169 of 2154

An upper-diagonal matrix expression for a serial kinematic chain can be derived from (6.43) and (6.45)

$$\begin{pmatrix} {}^1\boldsymbol{f}_1 \\ {}^2\boldsymbol{f}_2 \\ \vdots \\ {}^n\boldsymbol{f}_n \end{pmatrix} = \begin{pmatrix} {}^1\mathbf{X}_1^{\mathrm{F}1}\mathbf{A}_1 & {}^1\mathbf{X}_2^{\mathrm{F}2}\mathbf{A}_2 & \cdots & {}^1\mathbf{X}_n^{\mathrm{F}n}\mathbf{A}_n \\ \mathbf{0} & {}^2\mathbf{X}_2^{\mathrm{F}2}\mathbf{A}_2 & \cdots & {}^2\mathbf{X}_n^{\mathrm{F}n}\mathbf{A}_n \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & {}^n\mathbf{X}_n^{\mathrm{F}n}\mathbf{A}_n \end{pmatrix} \begin{pmatrix} \boldsymbol{\phi}_1 \\ \boldsymbol{\phi}_2 \\ \vdots \\ \boldsymbol{\phi}_n \end{pmatrix} . \tag{6.46}$$

This equation is linear in the unknown parameters, but the left-hand side is composed of a full force-torque vector at each joint. Since only the torque τ_i about the joint rotation axis z_i can usually be measured, each spatial force ${}^i\boldsymbol{f}_i$ must be projected onto the joint rotation axis, reducing (6.46) to

$$\boldsymbol{\tau} = \mathbf{K}\boldsymbol{\phi} \, , \tag{6.47}$$

where

$$\tau_i = \begin{pmatrix} z_i \\ \mathbf{0} \end{pmatrix} \cdot \boldsymbol{f}_i \, ,$$

$$\mathbf{K}_{ij} = \begin{pmatrix} z_i \\ \mathbf{0} \end{pmatrix} \cdot {}^i\mathbf{X}_j^{\mathrm{F}j}\mathbf{A}_j \, , \quad \boldsymbol{\phi} = \begin{pmatrix} \boldsymbol{\phi}_1 \\ \cdot \\ \cdot \\ \boldsymbol{\phi}_n \end{pmatrix}$$

and $\mathbf{K}_{ij} = \mathbf{0}_{1\times 10}$ if $i > j$. For an n-link manipulator, $\boldsymbol{\tau}$ is an $n\times 1$ vector, $\boldsymbol{\phi}$ is a $10n\times 1$ vector, and $\mathbf{K}$ is an $n\times 10n$ matrix.

For geared electric drives, joint torque may either be measured by joint torque sensors (Chap. 28) or estimated from the motor current by using an electric motor model (Chap. 8). Most robots do not have joint torque sensors, in which case joint friction needs to be taken into account. Joint friction typically consumes a large fraction of the torque that the motor produces. Coulomb and viscous friction are the most important components of a friction model, although Stribeck friction may need to be modeled for low joint velocities [6.24, 25]. A friction model is estimated through a process of moving one joint at a time, and relating motor torque to velocity. Ripple torque, due either to uncompensated inhomogeneities of the magnetic field in the motor [6.26, 27], or to positional dependencies of gear tooth interaction [6.24], may need to be modeled.

Equation 6.47 represents the dynamics of the manipulator for one sample point. It can be rewritten using P data points as with geometric calibration

$$\boldsymbol{\tau} = \mathbf{K}\boldsymbol{\phi} \, .$$

$$\mathbf{K} = \begin{pmatrix} \mathbf{K}^1 \\ \vdots \\ \mathbf{K}^P \end{pmatrix} , \quad \boldsymbol{\tau} = \begin{pmatrix} \boldsymbol{\tau}^1 \\ \vdots \\ \boldsymbol{\tau}^P \end{pmatrix} , \tag{6.48}$$

where $\boldsymbol{\tau}$ is now an $nP\times 1$ vector and $\mathbf{K}$ is now $nP\times 10n$.

Unfortunately, one cannot apply simple least-squares estimation because $\mathbf{K}^{\mathrm{T}}\mathbf{K}$ is not invertible due to the loss of rank from restricted degrees of freedom at the proximal links and the lack of full force-torque sensing. Some inertial parameters are completely unidentifiable, while some others can only be identified in linear combinations. The general topic of identifiability of parameters, and how to handle unidentifiable parameters, is discussed next.

An issue with geared electric drives is the rotor inertia. If not known, the rotor inertia can be added to the list of 10 inertial parameters to be identified for a link [6.28]. For large gear ratios, the rotor inertia can dominate the other components of link inertia.

6.3.2 Load Inertial Parameter Estimation

The load is considered as a rigid body held by the end effector. It has 10 inertial parameters: mass m_{L}, first moments $m_{\mathrm{L}}c_{\mathrm{L}}$ relative to O_n and symmetric inertia matrix $\bar{\mathbf{I}}_{\mathrm{L}}$ referred here to origin O_n (Fig. 6.11). The estimation of these parameters can be used to tune the control law parameters in order to improve the dynamic accuracy of the robot, or to compensate for its dynamics by the control. They can also be exploited to verify the load transported by the robot. Two procedures are presented to estimate the load inertia parameters: the first one assumes that there is a wrist-mounted 6-axis force/torque sensor, whereas the second one makes use of the joint torques.

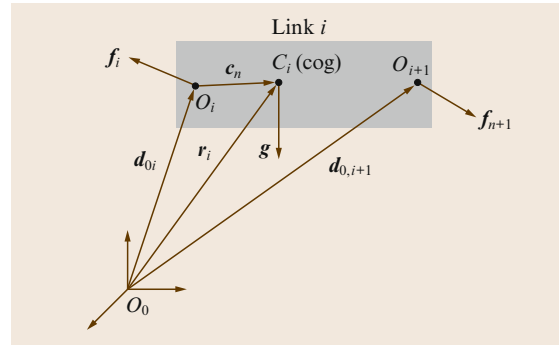


Fig. 6.11 Dynamics of the last link

Avidbots Ex1016
Page 170 of 2154

Using 6-Axis Forces/Torque Sensor

Kinematically, the force-torque sensor is part of the last link n, mounted near the joint axis z_n and origin O_n. The sensor provides readings relative to its own coordinate system, called $n+1$ (Fig. 6.11). The remainder of the last link is attached to the shaft of the force-torque sensor, and so the force-torque sensor is measuring the loads on the last link excluding itself. The acceleration of the force-torque sensor reference frame would have to be calculated based on the amount of offset from O_n, but we will ignore that complication here and assume that the sensor origin O_{n+1} is coincident with O_n.

The estimation of load inertial parameters can be carried out using (6.41).

The quantities inside the $\mathbf{A}_n$ matrix are computed by direct kinematics computation from the measured joint angles, and from the estimated joint velocities and accelerations. The estimation is typically done by bandpass filtering of the joint angle data [6.29]. The elements of the $\boldsymbol{f}_n$ vector are measured directly by the wrist force sensor. For robust estimates in the presence of noise, a larger number of P data points are obtained by moving the manipulator through a suitable trajectory. Augment $\boldsymbol{f}_n$ and $\mathbf{A}_n$ as

$$\mathbf{A} = \begin{pmatrix} \mathbf{A}_n^1 \\ \vdots \\ \mathbf{A}_n^P \end{pmatrix}, \quad \boldsymbol{f} = \begin{pmatrix} \boldsymbol{f}_n^1 \\ \vdots \\ \boldsymbol{f}_n^P \end{pmatrix}, \tag{6.49}$$

where P is the number of data points. The least-squares estimate for $\boldsymbol{\phi}_\mathrm{L}$ is given by

$$\hat{\boldsymbol{\phi}}_\mathrm{L} = (\mathbf{A}^\mathrm{T}\mathbf{A})^{-1}\mathbf{A}^\mathrm{T}\boldsymbol{f} \,. \tag{6.50}$$

For object recognition, it may be desired to derive the inertia about the center of mass, which can be done with the parallel axis theorem. An eigenvalue analysis can diagonalize the inertia matrix to reveal the principal axes and inertia.

Using Joint Torques

When the robot is holding a payload, the dynamic identification (6.48) can be rewritten as follows

$$\boldsymbol{\tau} = \mathbf{K}\boldsymbol{\phi} + \mathbf{K}_\mathrm{L}\boldsymbol{\phi}_\mathrm{L} \,, \tag{6.51}$$

where $\boldsymbol{\tau}$ contains the measured joint torques when the robot is holding a load for P points. Using (6.51) in the identification will result in grouping $\boldsymbol{\phi}_\mathrm{L}$ with $\boldsymbol{\phi}_n$ and with other link parameters. The load parameters can be deduced by examining how the load parameters have modified the values of the grouped inertial parameters using the expressions of grouping relations [6.30]. In the following we present three methods to identify $\boldsymbol{\phi}_\mathrm{L}$ parameters without using the grouping expressions.

Using a Priori Link Parameters Estimates. In this case we suppose that the robot parameters without load have been estimated as given in Sec. 6.3.1. By substituting the robot parameters into (6.51) the 10 inertial parameters of the load are identifiable and can be estimated using the least squares solution of [6.31]

$$(\boldsymbol{\tau} - \mathbf{K}\boldsymbol{\phi}) = \mathbf{K}_\mathrm{L}\boldsymbol{\phi}_\mathrm{L} \,. \tag{6.52}$$

In this method we suppose that the friction parameters are invariant with respect to the payload. If this hypothesis is not true the friction parameters must be re-estimated at the same step with $\boldsymbol{\phi}_\mathrm{L}$.

Using Joint Torques on the Same Trajectory Twice with and Without the Load. Since the vector $\mathbf{K}\boldsymbol{\phi}$ of (6.52) is equal to the joint torques without load, consequently, the payload inertial parameters could be identified by (6.52) after replacing $\mathbf{K}\boldsymbol{\phi}$ by $\boldsymbol{\tau}_0$ representing the joint torques on the same trajectory without load

$$(\boldsymbol{\tau} - \boldsymbol{\tau}_0) = \mathbf{K}_\mathrm{L}\boldsymbol{\phi}_\mathrm{L} \,. \tag{6.53}$$

This method assumes that the difference between the joint variables with and without payload is negligible and that the friction parameters are the same with and without load. If the friction parameters vary with respect to the load we can estimate this variation at the same time as the load inertial parameter.

Estimating the Robot Parameters and the Load Parameters in One Step. In this method the identification model is constructed by grouping two sets of equations: the first using a trajectory without load, and the second using a trajectory when the robot is holding the load. In this case the two sets of trajectories could be different. The link parameters and the payload inertial parameters are estimated using the least squares solution of

$$\begin{pmatrix} \boldsymbol{\tau}_a \\ \boldsymbol{\tau}_b \end{pmatrix} = \begin{pmatrix} \mathbf{K}_a & \mathbf{0} \\ \mathbf{K}_b & \mathbf{K}_\mathrm{L} \end{pmatrix} \begin{pmatrix} \boldsymbol{\phi} \\ \boldsymbol{\phi}_\mathrm{L} \end{pmatrix}, \tag{6.54}$$

where the indices a and b indicate the trajectories without and with load, respectively.

This method has the advantage of using a global identification procedure that can avoid the accumulation of errors of previous methods.

These three methods have been validated experimentally on the 6-DOF robot RX-90 of Staubli. They give similar results [6.30].

Avidbots Ex1016
Page 171 of 2154

6.3.3 Identification of Total Joint Drive Gains

For gear electric drive of joint j, the joint torque τ_j is estimated as $\tau_j = v_{rj} g_j$, where v_{rj} is the control signal computed by the numerical controller of the robot (the input references of the motor current loop) and g_j is the total joint j drive gain, given by the gear ratio, the current amplifier gain, and the motor torque constant.

Because of large values of the gear ratio for industrial robots, (> 50), joint drive gain is very sensitive to errors and must be accurately measured from special, time consuming, heavy tests, on the drive chain [6.32, 33].

The methods used to identify the load inertial parameters using joint torques can be inversed to identify the drive gains. In this case some load inertial parameters ϕ_{kL} must be known while the other unknown parameters ϕ_{uL} are identified with the gain $\boldsymbol{g}$.

The vector $\boldsymbol{\tau}$ of sampled joint torques in (6.52) or (6.54) is rewritten as

$$\boldsymbol{\tau} = \mathbf{V}_r \boldsymbol{g} \,, \tag{6.55}$$

where: $\boldsymbol{g}$ is the $(n \times 1)$ vector of the joint drive gains, $\mathbf{V}_r$ regroups the samples of v_{rj}.

Using the Same Set of Trajectory Twice, with and Without the Load

Introducing (6.55) and the known load inertial parameters in (6.53) with the expression

$$\mathbf{K}_L \phi_L = \mathbf{K}_{kL} \boldsymbol{\phi}_{kL} + \mathbf{K}_{uL} \boldsymbol{\phi}_{uL} \tag{6.56}$$

gives

$$(\mathbf{V}_r - \mathbf{V}_{ruL}) \boldsymbol{g} = \mathbf{K}_{kL} \boldsymbol{\phi}_{kL} + \mathbf{K}_{uL} \boldsymbol{\phi}_{uL} \,. \tag{6.57}$$

This equation can be used to identify $\boldsymbol{\phi}_{uL}$ and $\boldsymbol{g}$ as the *LS* solution of the system

$$[\mathbf{K}_{kL} \boldsymbol{\phi}_{kL}] = \begin{pmatrix} (\mathbf{V}_r - \mathbf{V}_{ruL}) & -\mathbf{K}_{uL} \end{pmatrix} \begin{pmatrix} \boldsymbol{g}^T & \boldsymbol{\phi}_{uL}^T \end{pmatrix}^T \,. \tag{6.58}$$

Using Two Sets of Trajectories (Same or Different), the First Set Without Load and the Second Set with the Load

Introducing (6.55) and the known load inertial parameters (6.56) in (6.54) gives

$$\begin{pmatrix} \mathbf{V}_{ra} \\ \mathbf{V}_{rb} \end{pmatrix} \boldsymbol{g} = \begin{pmatrix} \mathbf{K}_a & 0 & 0 \\ \mathbf{K}_b & \mathbf{K}_{uL} & \mathbf{K}_{kL} \end{pmatrix} \begin{pmatrix} \boldsymbol{\phi}^T & \boldsymbol{\phi}_{uL}^T & \boldsymbol{\phi}_{kL}^T \end{pmatrix}^T \,. \tag{6.59}$$

Equation (6.59) can be used to identify all the parameters, $\boldsymbol{g}$, $\boldsymbol{\phi}$ and $\boldsymbol{\phi}_{uL}$, as the single global *LS* solution of the system

$$\begin{pmatrix} 0 \\ \mathbf{K}_{kL} \boldsymbol{\phi}_{kL} \end{pmatrix} = \begin{pmatrix} \mathbf{V}_{ra} & -\mathbf{K}_a & 0 \\ \mathbf{V}_{rb} & -\mathbf{K}_b & -\mathbf{K}_{uL} \end{pmatrix} \begin{pmatrix} \boldsymbol{g}^T & \boldsymbol{\phi}^T & \boldsymbol{\phi}_{uL}^T \end{pmatrix}^T \,. \tag{6.60}$$

But owing to the fact that $\mathbf{K}_{kL}$ and $\mathbf{K}_{uL}$ are correlated by the same noisy data $(q, \dot{q}, \ddot{q})$, the total least squares (TLS) solution is used to avoid biais errors of the ordinary *LS* solution [6.34]. Rewritting (6.60) as

$$\mathbf{K}_{tot} \boldsymbol{\phi}_{tot} = \mathbf{0} \,, \tag{6.61}$$

where

$$\mathbf{K}_{tot} = \begin{pmatrix} \mathbf{V}_{ra} & -\mathbf{K}_a & 0 & 0 \\ \mathbf{V}_{rb} & -\mathbf{K}_b & -\mathbf{K}_{uL} & -\mathbf{K}_{kL} \boldsymbol{\phi}_{kL} \end{pmatrix} , \tag{6.62}$$

$$\boldsymbol{\phi}_{tot} = \begin{pmatrix} \boldsymbol{g}^T & \boldsymbol{\phi}^T & \boldsymbol{\phi}_{uL}^T & 1 \end{pmatrix}^T \,. \tag{6.63}$$

The singular value decomposition (*SVD*) of $\mathbf{K}_{tot}$ is given by

$$\mathbf{K}_{tot} = \mathbf{U} \mathbf{S} \mathbf{V}^T \,, \tag{6.64}$$

where $\mathbf{U}$ and $\mathbf{V}$ are orthonormal matrices, and $\mathbf{S} = \text{diag}(s_i)$ is a diagonal matrix with singular values s_i of $\mathbf{K}_{tot}$ sorted in decreasing order. There are infinity of vectors $\hat{\boldsymbol{\phi}}_{tot} = \lambda \mathbf{V}_c$ which are the *TLS* solutions of (6.61), depending on a scale factor λ and on the normalized *TLS* solution given by the last column $\mathbf{V}_c$ of $\mathbf{V}$, $||\mathbf{V}_c|| = 1$, corresponding to the smallest singular value s_c [6.34]. The unique solution $\hat{\boldsymbol{\phi}}_{tot}^*$ assumes that its last coefficient $\hat{\boldsymbol{\phi}}_{tot}^*(end)$ must be equal to 1 according to (6.63) such as

$$\hat{\boldsymbol{\phi}}_{tot}^* = \frac{\mathbf{V}_c}{\mathbf{V}_c(\text{end})} \tag{6.65}$$

The methods has been successfully experimentally validated on the 6-DOF Staubli TX-40 industrial robot [6.35].

This approach is very simple to perform and the experimental results have shown its effectiveness: for the global identification of the total joint drive gains and the dynamic parameters, it is only necessary to accurately weigh a payload mass and to carry standard trajectories of industrial robot.

6.3.4 Link Parameter Estimation for More Complex Structures

In this section we present the dynamic identification model for kinematic tree (spanning tree) robots and

Avidbots Ex1016
Page 172 of 2154

kinematic closed-loop robots including parallel robots. These models are linear in the inertial parameters and can be represented by a model similar to (6.47), thus these parameters can be identified using the same techniques.

Tree-Structured Robots

For an n-link manipulator with a kinematic tree structure, the dynamic identification model (6.46) must be modified to take into account that the inertial parameters of link j have no effect on the dynamics of links which do not belong to the chain from the base to that link. For such structures, the link on which link j is articulated is denoted by $p(j)$ (parent of j – see Sect. 3.4). It could be a link with any number i, such that $i < j$. Thus the nonzero elements of the column of the matrix representing the coefficients of the inertial parameters of link j in (6.46) would be

$$ {}^{j}\mathbf{X}_{j}^{\mathrm{F}j}\mathbf{A}_{j}, {}^{p(j)}\mathbf{X}_{j}^{\mathrm{F}j}\mathbf{A}_{j}, {}^{p(p(j))}\mathbf{X}_{j}^{\mathrm{F}j}\mathbf{A}_{j}, \ldots, {}^{b}\mathbf{X}_{j}^{\mathrm{F}j}\mathbf{A}_{j} , \qquad (6.66) $$

where b is the first link of the chain connecting link 0 to link j, thus $p(b) = 0$.

Consequently, in (6.47) of a tree robot we obtain: $\mathbf{K}_{ij} = \mathbf{0}_{1\times 10}$ if $i > j$ or if link i does not belong to the branch connecting link 0 to link j. This means that some elements of the upper-right submatrix of $\mathbf{K}$ will be zero.

We note that a serial structure is a particular case of a tree structure where $p(j) = j - 1$. Thus any link m such that $m < j$ will belong to the chain from the base to link j.

Closed-Loop Robots

The dynamic model of a closed-loop structure can be obtained by considering first an equivalent spanning tree by opening one joint at each closed loop, and then using the principle of virtual work such that

$$ \boldsymbol{\tau} = \mathbf{G}^{\mathrm{T}}\mathbf{K}_{\mathrm{tr}}\boldsymbol{\phi} , \quad \mathbf{G} = \left(\frac{\partial \boldsymbol{q}_{\mathrm{tr}}}{\partial \boldsymbol{q}_{\mathrm{a}}}\right) , \qquad (6.67) $$

with $\boldsymbol{q}_{\mathrm{a}}$ an $N \times 1$ vector of N active joint angles (N differs from n, the total number of joints), and $\boldsymbol{q}_{\mathrm{tr}}$ an $n \times 1$ vector of the joints of the spanning tree structure.

Parallel Robots

A parallel robot is a particular case of closed-loop robots (Chap. 18). It is composed of a moving platform, representing the terminal link, that is connected to the base by m parallel legs. The dynamic model of parallel robots can be given by [6.36]

$$ \boldsymbol{\tau} = \mathbf{J}^{\mathrm{T}}\mathbf{A}_{p}\boldsymbol{\phi}_{p} + \sum_{i=1}^{m}\left(\frac{\partial \boldsymbol{q}_{i}}{\partial \boldsymbol{q}_{\mathrm{a}}}\right)^{\mathrm{T}}\mathbf{K}_{i}\boldsymbol{\phi}_{i} , \qquad (6.68) $$

where $\mathbf{K}_i\boldsymbol{\phi}_i$ represents the dynamic model of leg i with $\mathbf{K}_i$ a function of $(\boldsymbol{q}_i, \dot{\boldsymbol{q}}_i, \ddot{\boldsymbol{q}}_i)$, $\boldsymbol{q}_i$ is the vector of joint angles of leg i, $\mathbf{A}_p\boldsymbol{\phi}_p$ is the Newton–Euler spatial force of the platform calculated in terms of the Cartesian variables of the mobile platform using (6.41), and $\mathbf{J}$ is the Jacobian matrix of the parallel manipulator. We can obtain (6.68) from (6.67) by supposing that the spanning tree structure is obtained by separating the mobile platform from the legs.

Equation 6.68 can be rewritten as

$$ \begin{aligned} \boldsymbol{\tau} &= \begin{bmatrix}\mathbf{J}^{\mathrm{T}}\mathbf{A}_{p} & \left(\partial \boldsymbol{q}_{1}/\partial \boldsymbol{q}_{\mathrm{a}}\right)^{\mathrm{T}}\mathbf{K}_{1} & \cdots & \left(\partial \boldsymbol{q}_{m}/\partial \boldsymbol{q}_{\mathrm{a}}\right)^{\mathrm{T}}\mathbf{K}_{m}\end{bmatrix} \\ &\quad \boldsymbol{\phi}_{\mathrm{par}} \\ \boldsymbol{\tau} &= \mathbf{K}_{\mathrm{par}}\left(\boldsymbol{\omega}_{p}, \dot{\boldsymbol{v}}_{p}, \dot{\boldsymbol{\omega}}_{p}, \boldsymbol{q}_{i}, \dot{\boldsymbol{q}}_{i}, \ddot{\boldsymbol{q}}_{i}\right)\boldsymbol{\phi}_{\mathrm{par}} , \end{aligned} \qquad (6.69) $$

where $\boldsymbol{\phi}_{\mathrm{par}}$ is the vector of the inertial parameters of the robot (the legs and the platform).

$$ \boldsymbol{\phi}_{\mathrm{par}} = \begin{pmatrix}\boldsymbol{\phi}_{p} \\ \boldsymbol{\phi}_{1} \\ \vdots \\ \boldsymbol{\phi}_{m}\end{pmatrix} . $$

In the common case where the legs are identical and their inertial parameters are denoted by $\boldsymbol{\phi}_{\mathrm{leg}}$, the identification model can be rewritten by the following equation, which considerably reduces the number of inertial parameters to be identified

$$ \boldsymbol{\tau} = \begin{bmatrix}\mathbf{J}^{\mathrm{T}}\mathbf{A}_{p} & \sum_{i=1}^{m}\left(\partial \boldsymbol{q}_{i}/\partial \boldsymbol{q}_{\mathrm{a}}\right)^{\mathrm{T}}\mathbf{K}_{i}\end{bmatrix}\begin{pmatrix}\boldsymbol{\phi}_{p} \\ \boldsymbol{\phi}_{\mathrm{leg}}\end{pmatrix} . \qquad (6.70) $$

The identification of the orthoglide parallel robot is given in [6.37].

6.4 Identifiability and Numerical Conditioning

That some inertial parameters in (6.47) are unidentifiable does not mean that they are intrinsically identifiable, only that the experimental setup does not allow them to be identified. Limited motion near the base could be fixed by placing the whole manipulator on a six-axis moving platform such as a Stewart–Gough platform. In fact, for mobile manipulators mounted on high-mobility vehicles such as satellites, it may be necessary to know the full inertial model of the manipulator. Additional sensors could be added, for ex-

Avidbots Ex1016
Page 173 of 2154

ample a six-axis force-torque sensor to the base of the robot [6.5], to identify some additional (but not all) inertial parameters.

A similar situation may arise in kinematic calibration, where for example joint models might be augmented to include gear eccentricities, transmission and coupling coefficients of gears, joint elasticity, link elasticity, and base deflection [6.38]. By instrumenting the robot with additional sensors, for example, by placing joint angle sensors before and after the gears to measure joint deflection, it may be possible to identify additional parameters of interest. Additional sensing with a theodolite was done to measure base deflection in [6.12].

Given that the experimental setup is what it is, there may be no recourse but to deal with unidentifiable parameters, treated in the next section. Another problem is that parameters might in principle be identifiable, but numerical conditioning prevents their accurate determination. Issues such as adequacy of the collected data, and the effect of different units and magnitudes of parameters, are treated subsequently.

6.4.1 Identifiability

There are two main approaches towards handling unidentifiable parameters, depending on whether the goal is a structural model or a prediction model. For a structural model, the goal is to find the minimum parameter set that provides a meaningful physical description of the system by eliminating parameters until all are identifiable. This is done by careful evaluation of the effect of each parameter of the original model.

For a prediction model, the goal is to match the outputs to the inputs as closely as possible, and is more of a curve-fitting approach. The resulting parameter values are not necessarily physically meaningful or accurate.

Structural Model

It may be possible in the initial model formation to avoid including redundant or unidentifiable parameters. At other times, it is not possible to determine a priori what the minimal parameter set is, because of system complexity or because of numerical difficulties such as measurement error or limitations in the collected data.

A Priori Parameter Elimination. In the initial model formation, the choice of representation can immediately dictate whether the model is redundant or minimal. For kinematic calibration, the minimal parameter set includes four parameters for a rotary joint and two parameters for a prismatic joint. The detailed presentation in Sect. 6.2.1 of how to set up the DH/Hayati parameters for different kinematic and sensing arrangements has the advantage that the parameter set is minimal (excepting joint models). Once done, the application to any manipulator is somewhat formulaic.

Five- and six-parameter joint models have also been proposed, either for the convenience of locating link coordinate systems or for the ease of model formation [6.13, 28, 39]. With such redundant parameter sets, extra steps must be taken to handle the numerical difficulties caused by the redundancy, such as by reducing the number of parameters in a postprocessing step. When there are complicated joint models that include gearing effects as mentioned above, the resultant large numbers of parameters make the parameter elimination problem more difficult. Determining the minimal set of inertial parameters offers a different sort of complexity, because large numbers of parameters are not identifiable or are identifiable only in linear combinations. Again, model reduction is one approach to handle this problem.

Two approaches towards determining a minimal or base parameter set are:

1. Numerical identification of the unidentifiable parameters or linear combinations of parameters.
2. Symbolic determination.

Numerical identification involves rank reduction through matrix manipulation of the regressor, either through QR decomposition or singular value decomposition [6.40]. If the kinematic or dynamic model is known exactly, then a simulation using exactly generated *data* will yield a noise-free regressor matrix $\mathbf{A}$ in (6.3). For QR decomposition, the regressor matrix is factored as

$$\mathbf{A} = \mathbf{Q}\begin{pmatrix}\mathbf{R}\\ \mathbf{0}_{MP-N_{\text{par}},N_{\text{par}}}\end{pmatrix}, \tag{6.71}$$

where $\mathbf{Q}$ is an $M \times MP$ orthogonal matrix, $\mathbf{R}$ is an $N_{\text{par}} \times N_{\text{par}}$ upper-triangular matrix, and $\mathbf{0}_{MP-N_{\text{par}},N_{\text{par}}}$ is the zero matrix with dimensions $MP - N_{\text{par}} \times N_{\text{par}}$. Theoretically, the unidentifiable parameters ϕ_i are those whose corresponding diagonal elements R_{ii} of matrix $\mathbf{R}$ are zero. In practice, $|R_{ii}|$ is considered to be zero if it is less than the numerical zero ζ [6.41]

$$\zeta = MP\epsilon \max_i |R_{ii}| , \tag{6.72}$$

where ϵ is the computer precision [6.41]. Other parameters may appear in linear combinations depending on how many nonzero elements there are in row j of $\mathbf{R}$. Resolving these linear combinations is arbitrary. One approach is to zero all elements in the linear combination save one. The result is a prediction model rather than a structural model.

Avidbots Ex1016
Page 174 of 2154

Symbolic determination of the base inertial parameters has been proposed in [6.40, 42]. Using an energy formulation, the total energy of link j is expressed as $\boldsymbol{h}_j\phi_j$, where $\boldsymbol{h}_j$ is a row vector termed the total energy function whose elements are kinematic expressions of the angular and linear velocity of link j plus gravity. A recursive relation between neighboring links is developed as

$$\boldsymbol{h}_j = \boldsymbol{h}_{j-1}{}^{j-1}\boldsymbol{\lambda}_j + \dot{q}_j\boldsymbol{\eta}_j \,, \tag{6.73}$$

where the elements of the 10×10 matrix ${}^{j-1}\boldsymbol{\lambda}_j$ are functions of the DH parameters defining frame j, and the elements of the 1×10 row vector $\boldsymbol{\eta}_j$ depend on the linear and angular velocities. The details of these expressions can be found in [6.40, 42]. Grouping rules are then developed to find the precise linear combinations of parameters.

The minimum inertial parameters for a tree structure can be obtained using a closed-form solution similar to the serial structure case [6.42].

The term $\mathbf{K}_{\mathrm{tr}}\boldsymbol{\phi}$ in (6.67) shows that the minimum parameters of the spanning tree structure can be used to calculate the dynamics of the closed structure, giving by this a first reduction and grouping of the inertial parameters. However extra parameters can be eliminated or grouped when considering the matrix $\mathbf{G}$. The case of parallelogram loops can be treated symbolically using closed-form relations [6.42]. For general closed loops, the minimum parameters must be determined using numerical methods such as QR decomposition [6.40].

In the case of parallel robots, we deduce from (6.68) that the minimum parameters of the legs can be used to calculate $\mathbf{K}_i\boldsymbol{\phi}_i$, giving a first reduction of the inertial parameters. However, some other parameters can be grouped with the parameters of the platform. The minimum parameters of the Gough–Stewart robot are given in [6.43].

Data-Driven Parameter Elimination. The singular value decomposition of the regressor matrix can show which parameters are unidentifiable, weakly identifiable, or identifiable only in linear combination. For N_{par} parameters, P data points, and M-dimensional output measurements at each data point, the regressor matrix $\mathbf{A}$ (6.3) or (6.7) can be decomposed as

$$\mathbf{A} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^{\mathrm{T}} \,, \tag{6.74}$$

where $\mathbf{U}$ is an $MP\times MP$ orthogonal matrix, $\mathbf{V}$ is an $N_{\mathrm{par}}\times N_{\mathrm{par}}$ orthogonal matrix, and $\boldsymbol{\Sigma}$ is the $MP\times N_{\mathrm{par}}$ matrix of singular values

$$\boldsymbol{\Sigma} = \begin{pmatrix} \mathbf{S} \\ \mathbf{0}_{MP-N_{\mathrm{par}},N_{\mathrm{par}}} \end{pmatrix} , \tag{6.75}$$

where $\mathbf{S} = \mathrm{diag}(\mu_1,\ldots,\mu_r,0,\ldots,0)$ is the $N_{\mathrm{par}}\times N_{\mathrm{par}}$ matrix of ordered singular values, with μ_1 the largest and μ_r the smallest nonzero singular value. There may be $N_{\mathrm{par}}-r$ zero singular values $\mu_{r+1}=\cdots=\mu_{N_{\mathrm{par}}}=0$.

Especially when complex joint models are assumed that include flexibility, backlash, and gear eccentricity, it is not clear that all parameters can be identified. Retaining poorly identifiable parameters will degrade the robustness of the calibration; such parameters are indicated by zero or very small singular values. The expansion of (6.7) in terms of (6.74) is

$$\Delta\boldsymbol{y} = \sum_{j=1}^{r} \mu_j(\boldsymbol{v}_j^{\mathrm{T}}\Delta\boldsymbol{\phi})\boldsymbol{u}_j \,, \tag{6.76}$$

where $\boldsymbol{u}_j$ and $\boldsymbol{v}_j$ are the j-th columns of $\mathbf{U}$ and $\mathbf{V}$. For zero or small singular values μ_j, the projection $\boldsymbol{v}_j^{\mathrm{T}}\Delta\boldsymbol{\phi}$ of the parameters onto the column vector $\boldsymbol{v}_j$ in general represents a linear combination of parameters. It is also possible that only a single parameter results from the projection.

To proceed, it is first necessary to scale the parameters and the output measurements so that the singular values are comparable. Scaling is discussed in Sect. 6.4.3. Small singular values signal that there are parameters that are poorly identifiable and that should be eliminated. By small singular values, *Schröer* [6.38] suggests the heuristic that the *condition number* of a well-conditioned regressor matrix should be less than 100

$$\kappa(\mathbf{A}) = \frac{\mu_1}{\mu_r} < 100 \,. \tag{6.77}$$

This heuristic derives from experience of their statistical community. If the condition number is above 100, the singular values are examined, beginning with the smallest one, which may be a zero singular value.

If the condition number is above 100, the linear sums (6.76) corresponding to the smallest singular value μ_r are examined. The elements of column $\boldsymbol{v}_r$ are in one-to-one correspondence with the elements of $\Delta\boldsymbol{\phi}$. If there is an element j of $\boldsymbol{v}_r$ that is much larger than the others, then the parameter ϕ_j corresponding to that column element is a candidate for elimination. This procedure will tend to pinpoint parameters that are totally unidentifiable. Isolating the largest element of $\boldsymbol{v}_r$ is once again only meaningful if the parameters have been previously scaled.

Once the parameter is eliminated, the condition number of the reduced regressor is computed again. The procedure iterates until the condition number of the regressor is below 100.

The previous procedure can be carried out using QR decomposition as in (6.71), by replacing ϵ representing

Avidbots Ex1016
Page 175 of 2154

the computer precision in (6.72) by a greater value function of the noise level.

Prediction Model

If there are several largest elements of $\boldsymbol{v}_r$ that are roughly comparable in magnitude, then these parameters may only be identifiable in linear combination. There will be the same number of singular values which are too small. By examining several columns $\boldsymbol{v}_j$ corresponding to the small singular values, these linear combinations can become apparent. The linear combinations can be resolved arbitrarily, i. e., all parameter elements in the linear combination save one can be set to zero. The result of zeroing some parameters is that the model is no longer a structural model, but rather a prediction model.

One may also proceed without first removing parameters. It can be shown by substituting the singular value decomposition that

$$(\mathbf{A}^{\mathrm{T}}\mathbf{A})^{-1}\mathbf{A}^{\mathrm{T}} = \mathbf{V}\left(\mathbf{S}^{-1}\ \mathbf{0}_{N_{\mathrm{par}},MP-N_{\mathrm{par}}}\right)\mathbf{U}^{\mathrm{T}}\,. \tag{6.78}$$

Consequently the solution to (6.8) can be expressed as [6.44]

$$\Delta\boldsymbol{\phi} = \sum_{j=1}^{N_{\mathrm{par}}} \frac{\boldsymbol{u}_j^{\mathrm{T}}\Delta\boldsymbol{y}}{\mu_j}\boldsymbol{v}_j\,. \tag{6.79}$$

One can see that poorly identifiable parameters corresponding to small singular values μ_j greatly perturb the estimates, because the weighting is $1/\mu_j$. The strategy is to remove their influence. If μ_j is zero or very small relative to the largest singular value μ_1, then set $1/\mu_j = 0$.

Parameters that cannot be identified well are simply ignored in this procedure, which automatically converges to an identifiable parameter set. The resulting parameters can then be used for the model. A disadvantage is that the resulting parameters may not correspond to real model parameters. A procedure to project these parameters on the real ones is given in [6.40].

Incorporating A Priori Parameter Estimates

Least squares treats parameter values as completely unknown, i. e., they could be anywhere in the range from $-\infty$ to $+\infty$. Yet often a fairly good initial estimate of the parameters is available, for example, from a manufacturer's specifications or in the case of a recalibration. It makes sense to incorporate such an a priori preference into the least-squares optimization [6.45].

Suppose there is a preference for a solution near $\boldsymbol{\phi} = \boldsymbol{\phi}_0$. Express the preference as $\mathbf{I}\boldsymbol{\phi} = \boldsymbol{\phi}_0$, where $\mathbf{I}$ is the identity matrix, and append it as additional rows to (6.3) that reflect this preference

$$\begin{pmatrix}\mathbf{A}\\ \mathbf{I}\end{pmatrix}\boldsymbol{\phi} = \begin{pmatrix}\boldsymbol{y}\\ \boldsymbol{\phi}_0\end{pmatrix}. \tag{6.80}$$

To proceed with the solution, we can treat $\boldsymbol{\phi}_0$ as a constant. Redefine the parameter vector as $\tilde{\boldsymbol{\phi}} = \boldsymbol{\phi} - \boldsymbol{\phi}_0$, which we expect to be close to zero. Then

$$\begin{pmatrix}\mathbf{A}\\ \mathbf{I}\end{pmatrix}\tilde{\boldsymbol{\phi}} = \begin{pmatrix}\tilde{\boldsymbol{y}}\\ \mathbf{0}\end{pmatrix}, \tag{6.81}$$

where $\tilde{\boldsymbol{y}} = \boldsymbol{y} - \mathbf{A}\boldsymbol{\phi}_0$. We may not know $\boldsymbol{\phi}_0$ perfectly well, so we add a weighting parameter λ to express the confidence in this value

$$\begin{pmatrix}\mathbf{A}\\ \lambda\mathbf{I}\end{pmatrix}\tilde{\boldsymbol{\phi}} = \begin{pmatrix}\tilde{\boldsymbol{y}}\\ \mathbf{0}\end{pmatrix}. \tag{6.82}$$

The larger is λ, the more confidence we have in our a priori estimate. The least-squares solution is therefore

$$\begin{aligned}\tilde{\boldsymbol{\phi}} &= \left(\begin{pmatrix}\mathbf{A}^{\mathrm{T}} & \lambda\mathbf{I}\end{pmatrix}\begin{pmatrix}\mathbf{A}\\ \lambda\mathbf{I}\end{pmatrix}\right)^{-1}\begin{pmatrix}\mathbf{A}^{\mathrm{T}} & \lambda\mathbf{I}\end{pmatrix}^{\mathrm{T}}\begin{pmatrix}\tilde{\boldsymbol{y}}\\ \mathbf{0}\end{pmatrix}\\ &= (\mathbf{A}^{\mathrm{T}}\mathbf{A} + \lambda^2\mathbf{I})^{-1}\mathbf{A}^{\mathrm{T}}\tilde{\boldsymbol{y}}\,.\end{aligned} \tag{6.83}$$

This solution is called *damped least squares*, where λ is the damping factor. Expanding this solution in terms of the singular value decomposition,

$$\tilde{\boldsymbol{\phi}} = \sum_{j=1}^{N_{\mathrm{par}}}(\boldsymbol{u}_j^{\mathrm{T}}\tilde{\boldsymbol{y}})\frac{\mu_j}{\mu_j^2+\lambda^2}\boldsymbol{v}_j\,. \tag{6.84}$$

Hence a very small μ_j is counteracted by the larger λ value; then the a priori information about parameter values dominates the information from the data, i. e., the data are ignored. Thus for damped least squares, no explicit action on the singular values is required, because the damping factor modifies the singular values. The solution will be perturbed over the normal least-squares solution by the magnitude of the λ choice.

Confidence Measure of Parameter Estimates

After calibration, an estimate $\hat{\mathbf{M}}$ of the covariance of the parameter estimates can be derived from the data [6.3]. Assume that the task variables $\Delta\boldsymbol{y}$ have previously been scaled for equal uncertainty, that there is no bias, and that the errors are uncorrelated. Then

$$\hat{\mathbf{M}} = \sigma^2(\mathbf{A}^{\mathrm{T}}\mathbf{A})^{-1}\,. \tag{6.85}$$

An estimate of the standard deviation σ after running the calibration procedure is obtained from the χ^2 statistic [6.44, 46]

$$\chi^2 = (\Delta\boldsymbol{y} - \mathbf{A}\Delta\hat{\boldsymbol{\phi}})^{\mathrm{T}}(\Delta\boldsymbol{y} - \mathbf{A}\Delta\hat{\boldsymbol{\phi}})\,. \tag{6.86}$$

Avidbots Ex1016
Page 176 of 2154

An unbiased estimator for σ^2 is $\hat{\sigma}^2 = \chi^2/\nu$, where $\nu = MP - N_{\text{par}}$ is called the statistical *degrees of freedom*; ν subtracts from the total number of measurements MP the number N_{par} of parameters estimated, since some of the measurements went into determining $\boldsymbol{\phi}$.

The estimate $\hat{\mathbf{M}}$ can be used as a basis for eliminating parameters, by choosing those with the largest variances.

6.4.2 Observability

The measurements that are taken will influence the accuracy of parameter estimation. In kinematic calibration, the quality of the pose set has been measured by an *observability index*. In inertial parameter estimation, the quality of the identification trajectory has been termed *persistent excitation* [6.47]. Regardless of whether data is collected statically as in kinematic calibration or dynamically as in inertial parameter estimation, the result is just a bunch of numbers that go into the regressor matrix, and so common terminology is appropriate.

In statistics, optimal experimental design theory has given rise to several data measures termed *alphabet optimalities* [6.48]. Some of the most prominent are the following:

- *A-optimality* minimizes the trace of $(\mathbf{A}^{\mathrm{T}}\mathbf{A})^{-1}$ to obtain regression designs.
- *D-optimality* maximizes the determinant of $(\mathbf{A}^{\mathrm{T}}\mathbf{A})^{-1}$.
- *E-optimality* maximizes the minimum singular value of $(\mathbf{A}^{\mathrm{T}}\mathbf{A})^{-1}$.
- *G-optimality* minimizes the maximum prediction variance, and does not have a simple expression in terms of singular values.

Although ties to the experimental design literature have not typically been made [6.49], several of the proposed observability indexes for robot calibration have an alphabet-optimality counterpart. A-optimality does not have a counterpart in the robot calibration literature, and conversely a few proposed observability indexes do not have an alphabet-optimality counterpart. In [6.49], it is proved that E-optimality and G-optimality are equivalent for exact design.

Borm et al. [6.50, 51] proposed an observability index (here termed O_1 and numbered consecutively below) that maximizes the product of all of the singular values

$$O_1 = \frac{\sqrt[r]{\mu_1 \cdots \mu_r}}{\sqrt{P}} \,. \tag{6.87}$$

This is similar to D-optimality. The rationale is that O_1 represents the volume of a hyperellipsoid in $\Delta\boldsymbol{y}$, defined by (6.7) when $\Delta\boldsymbol{\phi}$ defines a hypersphere, and the singular values represent the lengths of axes. Therefore maximizing O_1 gives the largest hyperellipsoid volume, and hence a good aggregate increase of the singular values. One can also derive O_1 from the well-known relation $\sqrt{\det(\mathbf{A}^{\mathrm{T}}\mathbf{A})} = \mu_1 \cdots \mu_r$.

Minimizing the condition number of $\mathbf{A}$ as a measure of observability has been proposed in [6.40, 52, 53]

$$O_2 = \frac{\mu_1}{\mu_r} \,. \tag{6.88}$$

O_2 measures the eccentricity of the hyperellipsoid rather than its size. The intermediate singular values are not considered to be that pertinent, because minimizing the condition number automatically makes all singular values become more similar in magnitude and makes the hyperellipsoid closer to a hypersphere.

Nahvi et al. [6.22] argue for maximizing the minimum singular value μ_r as an observability measure

$$O_3 = \mu_r \,. \tag{6.89}$$

This is similar to E-optimality. The rationale is to make the shortest axis as long as possible, regardless of the other axes, that is to say, to improve the worst case. Consider the following standard result [6.4]

$$\mu_r \le \frac{\|\Delta\boldsymbol{y}\|}{\|\Delta\boldsymbol{\phi}\|} \le \mu_1 \,, \tag{6.90}$$

or more particularly

$$\mu_r \|\Delta\boldsymbol{\phi}\| \le \|\Delta\boldsymbol{y}\| \,. \tag{6.91}$$

Then maximizing μ_r ensures that a given size of parameter errors $\|\Delta\boldsymbol{\phi}\|$ has the largest possible influence on the pose errors $\|\Delta\boldsymbol{y}\|$.

Nahvi and *Hollerbach* [6.54] proposed the noise amplification index O_4, which can be viewed as combining the condition number O_2 with the minimum singular value O_3

$$O_4 = \frac{\sigma_r^2}{\sigma_1} \,. \tag{6.92}$$

The rationale is to measure both the eccentricity of the ellipse through O_2 as well as the size of the ellipse through O_3. The noise amplification index was argued to be the most sensitive to measurement error and modeling error.

Hollerbach and *Lokhorst* [6.21] found experimentally that the condition number and the minimum singular value gave about the same good result: their

Avidbots Ex1016
Page 177 of 2154

relative magnitudes were almost proportional to the root-mean-square (RMS) errors of the final parameter errors. The observability index O_1 was not as sensitive and not directly related to parameter errors. *Sun* and *Hollerbach* [6.49] derived general relations among the observability indexes and alphabet optimalities

$$O_1 \geq \text{A-optimality} \geq O_3 \,. \tag{6.93}$$

They further showed that if $\mu_1 \geq 1$, then $O_3 \geq O_2$, and if $\mu_r \leq 1$, then $O_2 \geq O_4$. They also argued that O_3 (D-optimality) is in general the best index, because it minimizes the variance of the parameters and also minimizes the uncertainty of the end-effector pose.

Optimal Experimental Design

An observability index is typically used to decide how many data points to collect. As one begins to add data points, the observability increases, then plateaus. Adding additional data does not then improve the quality of the estimates. For kinematic calibration, data may be added through random selection, or optimal design methods can be employed that can drastically reduce the amount of data required [6.55, 56]. Optimal experimental designs work by measuring the effect of adding or exchanging data points [6.57].

Exciting Trajectories

For inertial parameter estimation, data points are not independent because they result from a trajectory rather than isolated poses. Therefore the issue is what type of trajectory to generate. Industrial robots usually have joint position point-to-point trajectory generators. A continuous and smooth trajectory is calculated by interpolating between these points, assuming zero initial and final velocity and acceleration at each point, and using polynomial interpolators. Excitation trajectories are obtained by minimizing an observability index, using nonlinear optimization techniques to calculate the polynomial coefficients, under the constraints of the joint positions, velocities, and accelerations limits [6.52].

It is possible to facilitate the optimization by proceeding to a sequential excitation procedure. Specific trajectories, which structurally excite a small number of parameters, are easier to optimize. For example, moving one joint at a time with constant velocities excites friction and gravity parameters. In this approach, sequential identification is avoided. However, it is better to collect all the data and to proceed to a globally weighted least-squares estimation [6.29]. This procedure avoids the accumulation of estimation errors and allows the calculation of the confidence intervals (6.85).

Specific trajectories have been proposed like sinusoidal interpolators [6.58], or periodic trajectories obtained from a spectral analysis of the contribution function of the parameters [6.59]. This is a general planning trajectory strategy, which is very important to get accurate experimental identification [6.60].

6.4.3 Scaling

The numerical conditioning of parameter estimation can be improved by scaling both the output measurements (task variable scaling) and the parameters.

Task Variable Scaling

When performing a least-squares analysis on the endpoint pose error, position errors and orientation errors have to be combined (6.22)

$$||\Delta \boldsymbol{y}^i||^2 = ||\Delta^{-1} \boldsymbol{p}^l_{n+1}||^2 + ||\Delta^{-1} \boldsymbol{\rho}^l_{n+1}||^2 \,. \tag{6.94}$$

However, position error and orientation error have different units, and so are not comparable. Furthermore, not all position or orientation components may be measured with equal accuracy.

Ordinary least squares (6.8) weighs all task variables equally. To weigh these variables differently, the general solution is left multiplication of (6.7) by a scaling matrix $\mathbf{G}$ [6.45],

$$\begin{aligned} \mathbf{G}\Delta \boldsymbol{y} &= \mathbf{G}\mathbf{A}\Delta \boldsymbol{\phi} \,, \\ \Delta \tilde{\boldsymbol{y}} &\equiv \tilde{\mathbf{A}}\Delta \boldsymbol{\phi} \,, \end{aligned} \tag{6.95}$$

where $\Delta \tilde{\boldsymbol{y}} = \mathbf{G}\Delta \boldsymbol{y}$ is the scaled output vector and $\tilde{\mathbf{A}} = \mathbf{G}\mathbf{A}$ is the scaled regressor matrix. The weighted least-squares solution is

$$\Delta \boldsymbol{\phi} = (\tilde{\mathbf{A}}^{\mathrm{T}}\tilde{\mathbf{A}})^{-1}\tilde{\mathbf{A}}^{\mathrm{T}}\Delta \tilde{\boldsymbol{y}} = (\mathbf{A}^{\mathrm{T}}\mathbf{W}\mathbf{A})^{-1}\mathbf{A}^{\mathrm{T}}\mathbf{W}\Delta \boldsymbol{y} \,, \tag{6.96}$$

where $\mathbf{W} = \mathbf{G}^{\mathrm{T}}\mathbf{G}$. Often $\mathbf{W}$ is a diagonal matrix.

One approach to scaling position error versus orientation error is to equalize the effect of a parameter error $\Delta\phi_i$ on either position error or orientation error. Curiously, for human-sized robot arms the effect is equal without scaling due to an accident of metric units. If θ is the joint angle resolution, then $s = r\theta$ is the resulting endpoint position resolution. For human-sized arms, $r = 1$ m and hence $s = \theta$. Thus meters and radians are directly comparable after all. Thus using no scaling factors for the pose parameters makes some sense, and may explain why the general disregard for scaling in the robotics community has not had more consequences. If linkages are much smaller (like fingers) or much larger (like excavators) then the situation is different.

Avidbots Ex1016
Page 178 of 2154

A more general way of choosing the weighting matrix $\mathbf{W}$ is to use a priori information about acceptable relative errors. Such information might arise from specifications of the measurement apparatus. Suppose that the output variables are subject to independent Gaussian noise, such that σ_j^y is the standard deviation in task variable measurement component Δy_j^i, for $j = 1, \ldots, m$ pose components. Then the individual diagonal weights are $w_{jj} = 1/\sigma_j^y$, and define

$$\mathbf{R}^i = \text{diag}\left[\left(\sigma_1^y\right)^2, \ldots, \left(\sigma_m^y\right)^2\right] ,$$
$$\mathbf{R} = \text{diag}(\mathbf{R}^1, \ldots, \mathbf{R}^P) ,$$

where the weighting matrix $\mathbf{W} = \mathbf{R}^{-1}$, and $\mathbf{R}$ is called the *covariance matrix*.

The weighted least-squares estimate is

$$\Delta\boldsymbol{\phi} = (\mathbf{A}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{A})^{-1}\mathbf{A}^{\mathrm{T}}\mathbf{R}^{-1}\Delta\boldsymbol{y} . \tag{6.97}$$

The resulting scaled output variable $\Delta\tilde{y}_j^l = \Delta y_j^l/\sigma_j^y$ is dimensionless. The larger the uncertainty σ_j^y, the less this variable influences the least-squares solution relative to other variables. The standard deviations σ_j^y are not necessarily the same as endpoint measurement accuracy, because model errors and input noise also contribute to output errors.

The weighted least-squares solution using standard deviations is variously called the Gauss–Markov estimate, the generalized least-squares estimate, or the best linear unbiased estimator (BLUE) [6.3]. It is the minimum covariance estimate (on the parameter error) of all unbiased estimators. A significant point is that the standard deviations of the scaled components of $\Delta\tilde{\boldsymbol{y}}$ are all about the same size, or the covariance $\tilde{\mathbf{R}} = \text{cov}(\Delta\tilde{\boldsymbol{y}}) = \mathbf{I}$, the identity matrix. Hence the Euclidean norm of the error vector $\Delta\tilde{\boldsymbol{y}}$ is a reasonable measure of its size.

Often we do not know the covariance matrix $\mathbf{R}$ that well. An estimate of the standard deviations after running the calibration procedure is obtained from the χ^2 statistic [6.44, 46]

$$\chi^2 = (\Delta\boldsymbol{y} - \mathbf{A}\Delta\boldsymbol{\phi})^{\mathrm{T}}\mathbf{R}^{-1}(\Delta\boldsymbol{y} - \mathbf{A}\Delta\boldsymbol{\phi}) . \tag{6.98}$$

This equation is the same as the residual error (6.95), with substitution for $\mathbf{W} = \mathbf{R}^{-1}$. χ^2 is nothing more than the weighted residuals after calibration. The expected value of χ^2 is

$$E(\chi^2) \equiv \nu = PK - R , \tag{6.99}$$

where E is the expectation operator. That is to say, the unweighted residuals $(\Delta\boldsymbol{y} - \mathbf{A}\Delta\boldsymbol{\phi})^2$ should, with enough measurements, approximate the true covariance. We may now uniformly scale an initial estimate of $\mathbf{R}$ after a preliminary calibration, based on the value of χ^2,

$$\hat{\mathbf{R}} = \frac{\chi^2}{\nu}\mathbf{R} , \tag{6.100}$$

where $\hat{\mathbf{R}}$ is the revised estimate of the covariance matrix.

Parameter Scaling

Scaling of parameters is important for proper convergence in nonlinear optimization and for singular value decomposition. If parameters have vastly different magnitudes, then the singular values are not directly comparable. Also, parameter scaling can improve the conditioning of the regressor $\mathbf{A}$ and help to avoid invertibility problems.

Whereas left multiplication of $\mathbf{A}$ in (6.95) results in task variable scaling, right multiplication of $\mathbf{A}$ by a weighting matrix $\mathbf{H}$ results in parameter scaling [6.45]

$$\Delta\boldsymbol{y} = (\mathbf{AH})(\mathbf{H}^{-1}\Delta\boldsymbol{\phi}) \equiv \bar{\mathbf{A}}\Delta\tilde{\boldsymbol{\phi}} , \tag{6.101}$$

where the scaled Jacobian and parameters are $\bar{\mathbf{A}} = \mathbf{AH}$ and $\Delta\tilde{\boldsymbol{\phi}} = \mathbf{H}^{-1}\Delta\boldsymbol{\phi}$. The least-squares solution is not changed by parameter scaling, whereas it is by task variable scaling.

The most common approach towards parameter weighting is column scaling, which does not require a priori statistical information. Define a diagonal matrix $\mathbf{H} = \text{diag}(h_1, \ldots, h_{N_{\text{par}}})$ with elements

$$h_j = \begin{cases} \|\boldsymbol{c}_j\|^{-1} & \text{if } \|\boldsymbol{c}_j\| \neq 0 \\ 1 & \text{if } \|\boldsymbol{c}_j\| = 0 \end{cases} \tag{6.102}$$

where $\boldsymbol{a}_j$ is the j-th column of $\mathbf{A}$. Then (6.101) becomes

$$\Delta\boldsymbol{y} = \sum_{j=1}^{N_{\text{par}}} \frac{\boldsymbol{a}_j}{\|\boldsymbol{a}_j\|}\Delta\phi_j\|\boldsymbol{a}_j\| . \tag{6.103}$$

Suppose that $\Delta\boldsymbol{y}$ has been previously normalized; then its length is meaningful. Each $\boldsymbol{a}_j/\|\boldsymbol{a}_j\|$ is a unit vector, and hence each scaled parameter $\Delta\phi_j\|\boldsymbol{a}_j\|$ is about the same size and has the same effect on $\Delta\boldsymbol{y}$.

Schröer [6.38] identified a problem with column scaling, namely that poor identifiability of parameters can result in very small Euclidean norms, which result in very large scaling factors. This results in strong amplification of uncertainties of $\mathbf{A}$. Instead, Schröer proposed scaling based on the effect on the anticipated error of the robot (as previously discussed under task variable scaling).

Avidbots Ex1016
Page 179 of 2154

In an ideal case, one would have a priori knowledge of the expected value $\boldsymbol{\phi}_0$ of the parameter vector and the standard deviation σ_j^{ϕ} of each of the parameter vector components. More generally, the parameter distribution would be described by a covariance matrix $\mathbf{M}$, but usually this information is not available. Instead, the estimate $\hat{\mathbf{M}}$ of the covariance from (6.85) can be used.

If the output measurement covariance $\mathbf{R}^{-1}$ and parameter error covariance $\mathbf{M}$ are both available, one may define a new least-squares optimization criterion that combines output errors with parameter errors to yield a new χ^2 statistic

$$\chi^2 = (\Delta \boldsymbol{y} - \mathbf{A}\Delta\boldsymbol{\phi})^{\mathrm{T}}\mathbf{R}^{-1}(\Delta \boldsymbol{y} - \mathbf{A}\Delta\boldsymbol{\phi}) + \Delta\boldsymbol{\phi}^{\mathrm{T}}\mathbf{M}^{-1}\Delta\boldsymbol{\phi} \,. \tag{6.104}$$

Its solution is the minimum-variance estimate, which unlike (6.97) is biased

$$\Delta\boldsymbol{\phi} = (\mathbf{A}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{A} + \mathbf{M}^{-1})^{-1}\mathbf{A}^{\mathrm{T}}\mathbf{R}^{-1}\Delta\boldsymbol{y} \,. \tag{6.105}$$

The Kalman filter recursively solves the same problem [6.61, 62]; when the state is nonvarying, there is a constant process, and there is no process noise [6.13, 63]. The Gauss–Markov estimate is the limiting case when $\mathbf{M}^{-1}$ is zero, i. e., there is no a priori information about the parameters. Again, there is an issue of determining the covariances. As for the Gauss–Markov estimate, the expected value of χ^2 can be used to uniformly scale $\mathbf{R}$ and $\mathbf{M}$ *post facto* [6.64].

6.4.4 Recursive Least Squares and the Kalman Filter

In a control setting, it may be necessary to update the estimate $\hat{\boldsymbol{\phi}}$ at time step k based upon all the readings up to time step $k-1$. It is possible to recast least squares recursively to perform this updating [6.3], which is strongly analogous to the Kalman Filter. At time step k, there is a new reading

$$\boldsymbol{y}^k = \mathbf{A}^k\,\boldsymbol{\phi} + \boldsymbol{e}^k \,, \tag{6.106}$$

where $\boldsymbol{y}^k$ is of dimension $M \times 1$, $\mathbf{A}^k$ is of dimension $M \times N$, and $\boldsymbol{\phi}$ is of dimension $N \times 1$. Let $\mathbf{R}^k$ be the covariance of $\boldsymbol{e}^k$. This equation is stacked for all readings up to k, and the Gauss–Markov estimate is derived

$$\begin{aligned} \boldsymbol{y}_k &= \mathbf{A}_k\boldsymbol{\phi} + \boldsymbol{e}_k \\ \hat{\boldsymbol{\phi}}_k &= (\mathbf{A}_k^T\mathbf{R}_k^{-1}\mathbf{A}_k)^{-1}\mathbf{A}_k^T\mathbf{R}_k^{-1}\boldsymbol{y}_k \,, \end{aligned} \tag{6.107}$$

where $\boldsymbol{y}_k$ is of dimension $MK \times 1$, and $\mathbf{A}_k$ is of dimension $MK \times N$. $\mathbf{R}_k$ is the covariance of $\boldsymbol{e}_k$, and is a block-diagonal matrix with elements $\mathbf{R}^k$.

To make an analogy to the Kalman filter, define the covariance of $\hat{\boldsymbol{\phi}}$ as $\mathbf{P}_k = (\mathbf{A}_k^T\mathbf{R}_k^{-1}\mathbf{A}_k)^{-1}$ of dimension $N \times N$. Inverting this relation eventually yields a recursive reltionship in terms of the covariance up to the previous time, step, plus a new contribution

$$\mathbf{P}_k^{-1} = \mathbf{P}_{k-1}^{-1} + (\mathbf{A}^k)^T(\mathbf{R}^k)^{-1}\mathbf{A}^k \,. \tag{6.108}$$

Similarly,

$$\mathbf{A}_k^T\mathbf{R}_k^{-1}\boldsymbol{y}_k = \mathbf{A}_{k-1}^T\mathbf{R}_{k-1}^{-1}\boldsymbol{y}_{k-1} + (\mathbf{A}^k)^T(\mathbf{R}^k)^{-1}\boldsymbol{y}^k \,. \tag{6.109}$$

Substituting the definition of $\mathbf{P}_k$ and (6.109) into (6.107) eventually yields

$$\hat{\boldsymbol{\phi}}_k = \hat{\boldsymbol{\phi}}_{k-1} + \mathbf{P}_k(\mathbf{A}^k)^T(\mathbf{R}^k)^{-1}(\boldsymbol{y}^k - \mathbf{A}^k\,\hat{\boldsymbol{\phi}}_{k-1}) \,. \tag{6.110}$$

In the covariance updating matrix (6.108), the inverse $\mathbf{P}_{k-1}^{-1}$ has to be taken. The updating relationship can be reworked into a computationally friendlier form using the binomial inverse theorem (see Wikipedia entry)

$$(\mathbf{A} + \mathbf{UBV})^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{U}(\mathbf{B}^{-1} + \mathbf{VA}^{-1}\mathbf{U})^{-1}\mathbf{VA}^{-1} \tag{6.111}$$

Making the identification $\mathbf{A} = \mathbf{P}_{k-1}^{-1}$, $\mathbf{U} = (\mathbf{A}^k)^T$, $\mathbf{B} = (\mathbf{R}^k)^{-1}$, and $\mathbf{V} = \mathbf{A}^k$, after inversion (6.108) becomes

$$\mathbf{P}_k = \mathbf{P}_{k-1} - \mathbf{P}_{k-1}(\mathbf{A}^k)^T(\mathbf{R}^k + \mathbf{A}^k\mathbf{P}_{k-1}(\mathbf{A}^k)^T)^{-1}\mathbf{A}^k\,\mathbf{P}_{k-1} \tag{6.112}$$

The resulting matrix in parentheses is of size $M \times M$, and is typically of much lower dimension than the $N \times N$ dimension of $\mathbf{P}_k$. That is to say, there are usually many more parameters than measurement components. Hence the inversion is easier.

Equations (6.110) and (6.112) comprise a large part of the Kalman filter. Thus the Kalman filter is fundamentally recursive least squares [6.3, 61, 62]. There are many variants to the Kalman filter, including one based on the mmse estimate.

Gautier and *Poignet* [6.65] applied an extended Kalman filter to estimate a robot's inertial parameters, in which the filter was reworked to include velocity and acceleration estimation. They found that there was no advantage to using a Kalman filter over least squares, and in fact there was a disadvantage in terms of sensitivity to initial conditions and slower

convergence. One issue in on-line versus off-line estimation is the accuracy of velocity and acceleration estimates. In off-line estimation, two sided (or acausal) filters can be used, and they typically estimate time derivatives more accurately than one-side (or causal) filters.

6.5 Conclusions and Further Reading

This chapter has presented methods for calibrating the kinematic parameters and the inertial parameters of robot manipulators. Both methods are instances of parameter estimation using least squares. Inertial parameters appear linearly in the equations of motion, and ordinary least squares can be used. Kinematic parameters appear nonlinearly because of sines and cosines, and so nonlinear estimation via the Gauss–Newton method can be applied.

There are domain-specific issues for setting up the calibration equations. For kinematic calibration, Hayati parameters have to be mixed with Denavit–Hartenberg parameters to handle the case of nearly parallel joint axes. The calibration equations have to take into account how the endpoint is measured or constrained. By examining in a detailed fashion the possible joint sequences, including parallel or prismatic joints, a minimal parameterization is achieved that obviates problems of identifiability.

The calibration index has been presented as categorizing all kinematic calibration methods. This index counts the number of equations generated per pose, by calculating the excess of sensing over mobility. A key is that all calibration methods can be viewed as closed-loop methods, where any endpoint sensing system is considered as a joint. Parallel robots are handled by incorporating multiple closed loops.

For inertial parameter estimation,the recursive Newton–Euler equations lead to a formulation of the regressor matrix as an upper-diagonal matrix. A minimal parameterization is straightforward for serial and spanning robots. The load inertial parameters estimation has been treated by using either the joint torques or the terminal link force-torque sensor measurements. The proposed models are exploited to identify the joint torques gains supposing known load parameters.

Numerical methods were presented to handle unidentifiable parameters if a minimal parameterization has not been achieved. These methods hinge on the singular value or QR decomposition of the regressor matrix. The singular values can be examined to decide which parameters are not identifiable and should be eliminated. Alternatively, a small singular value can simply be zeroed to eliminate the effects of poorly identifiable parameters without explicitly eliminating them. The former yields a structural model, while the latter yields a prediction model. A priori estimates of parameters can be taken into account by using damped least squares.

The adequacy of the measurement set for parameter estimation is captured by different proposals for an observability index. This index can be related to the literature on alphabet optimalities in experimental design.

Finally, the scaling of measurements or of parameters is important for a well-conditioned numerical estimation, and is essential in order to compare singular values. When uncertainties in measurements and in parameters are included as weights, the optimal minimum variance estimate can be found, which is related to the Kalman filter. If not known beforehand, these uncertainties can be estimated from the data.

6.5.1 Relation to Other Chapters

Estimation involving least squares and Kalman filtering is discussed in Chap. 5 in a similar context. The problem of estimating properties of the world through sensors is very similar to model identification. Recursive estimation techniques are particularly appropriate when a robot needs to update its world model incrementally. For model identification, employing a recursive formulation is not particularly helpful, as the machinery of recursive updating can obscure numerical issues with the total data.

The singular value decomposition appears in other contexts. Chapter 10 and 16 analyze equal motion capability in different directions by measures similar to the observability indexes: O_1 has a counterpart in the manipulability measure, and O_2 the condition number and O_3 the minimum singular value appear again. By contrast, the concern in calibration is good data in all directions as captured by the singular values. Chap. 10 employs the singular value decomposition to analyze redundant mechanisms. Whereas parameter estimation generally is an overconstrained least-squares problem (many more measurements than parameters), redundant mechanisms are underconstrained (more joint angles than task variables). Instead of signalling identifiability problems, zero singular values indicate the null space of the Jacobian. Damped least squares is used in Chap. 10 to avoid singularities. Just as the true parameters are perturbed by damped least squares in calibration, the

Avidbots Ex1016
Page 181 of 2154

trajectory is perturbed in order to get around a numerical conditioning issue.

Sensors involved in positioning a robot can have aspects of their sensor models calibrated as well, such as the gain of a potentiometer. Camera calibration is discussed in Chap. 32. Camera models can be determined at the same time as the kinematic models of robot manipulators [6.39, 66], including intrinsic camera parameters such as the pinhole model discussed in Chap. 5 as well as the extrinsic parameters of where the camera is located.

6.5.2 Further Reading

Screw Axis Measurement

An alternative to nonlinear least squares to estimate kinematic parameters is a class of methods which measure the joint axes as lines in space, termed *screw axis measurement* in [6.8]. One approach is *circle point analysis*, which involves moving one joint at a time to generate a circle at a distal measurement point [6.13]. Other approaches measure the Jacobian matrix, which contains as columns the joint screws [6.67]. With knowledge of the joint axes, the kinematic parameters are straightforwardly extracted without nonlinear search. The accuracy of this class of methods may not be as good as the nonlinear least-squares methods.

Total Least Squares

Ordinary least squares assumes that there is only noise in the output measurements, but often there is noise in the inputs as well. Input noise is known to lead to bias errors [6.3]. A framework for handling both input and output noise is total least squares [6.34], also known as orthogonal distance regression [6.68] or errors-in-variables regression [6.69]. Nonlinear total least squares has been applied to robot calibration in [6.2, 70, 71]. In the implicit loop method of [6.2], by treating endpoint measurements equally with joint measurements, no distinction is made between input and output noise.

Methods with Direct and Inverse Dynamic Models

The identification of dynamic parameters presented in this chapter is based on the Inverse Dynamic Identification Model and linear LS techniques. A common procedure to validate the identification result is to simulate the direct dynamic model with the identified parameters and compare the output with the real system.

Recently two methods which use both the direct and the inverse dynamic models have been developed to improve the noise immunity of estimates with respect to low rate or corrupted data in the observation matrix resulting from noisy measurements and/or bad tuning of joint positions band-pass filtering. The first method needs only the joint torques, while the velocity and acceleration are obtained from the direct dynamic model. It carries out in a combined procedure the identification and the validation. Current work aims to use this technique to identify robots with flexible joints where the flexible position variables are not measurable. The second method uses an instrumental variable (IV) approach. Both methods have been validated on a 6-DOF industrial robot [6.72, 73].

Video-References

VIDEO 422 Calibration of ABB's IRB 120 industrial robot
available from http://handbookofrobotics.org/view-chapter/06/videodetails/422

VIDEO 425 Robot calibration using a touch probe
available from http://handbookofrobotics.org/view-chapter/06/videodetails/425

VIDEO 430 Calibration and accuracy validation of a FANUC LR Mate 200iC industrial robot
available from http://handbookofrobotics.org/view-chapter/06/videodetails/430

VIDEO 480 Dynamic identification of Staubli TX40: Trajectory without load
available from http://handbookofrobotics.org/view-chapter/06/videodetails/480

VIDEO 481 Dynamic identification of Staubli TX40: Trajectory with load
available from http://handbookofrobotics.org/view-chapter/06/videodetails/481

VIDEO 482 Dynamic identification of Kuka LWR: Trajectory without load
available from http://handbookofrobotics.org/view-chapter/06/videodetails/482

VIDEO 483 Dynamic identification of Kuka LWR: Trajectory with load
available from http://handbookofrobotics.org/view-chapter/06/videodetails/483

VIDEO 485 Dynamic identification of a parallel robot: Trajectory with load
available from http://handbookofrobotics.org/view-chapter/06/videodetails/485

VIDEO 486 Dynamic identification of Kuka KR270: Trajectory without load
available from http://handbookofrobotics.org/view-chapter/06/videodetails/486

Avidbots Ex1016
Page 182 of 2154

VIDEO 487 Dynamic identification of Kuka KR270: trajectory with load
available from http://handbookofrobotics.org/view-chapter/06/videodetails/487
VIDEO 488 Dynamic identification of a parallel robot: Trajectory without load
available from http://handbookofrobotics.org/view-chapter/06/videodetails/488

References

6.1 P.Z. Marmarelis, V.Z. Marmarelis: *Analysis of Physiological Systems* (Plenum, London 1978)
6.2 C.W. Wampler, J.M. Hollerbach, T. Arai: An implicit loop method for kinematic calibration and its application to closed-chain mechanisms, IEEE Trans. Robotics Autom. **11**, 710–724 (1995)
6.3 J.P. Norton: *An Introduction to Identification* (Academic, London 1986)
6.4 G.H. Golub, C.F. Van Loan: *Matrix Computations* (Johns Hopkins Univ. Press, Baltimore 1989)
6.5 H. West, E. Papadopoulos, S. Dubowsky, H. Cheah: A method for estimating the mass properties of a manipulator by measuring the reaction moments at its base, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (1989) pp. 1510–1516
6.6 R.P. Paul: *Robot Manipulators: Mathematics, Programming, and Control* (MIT Press, Cambridge 1981)
6.7 S.A. Hayati, M. Mirmirani: Improving the absolute positioning accuracy of robot manipulators, J. Robotics Syst. **2**, 397–413 (1985)
6.8 J.M. Hollerbach, C.W. Wampler: The calibration index and taxonomy of kinematic calibration methods, Int. J. Robotics Res. **15**, 573–591 (1996)
6.9 A. Goswami, A. Quaid, M. Peshkin: Identifying robot parameters using partial pose information, IEEE Control Syst. **13**, 6–14 (1993)
6.10 M.R. Driels, W.E. Swayze: Automated partial pose measurement system for manipulator calibration experiments, IEEE Trans. Robotics Autom. **10**, 430–440 (1994)
6.11 G.-R. Tang, L.-S. Liu: Robot calibration using a single laser displacement meter, Mechatronics **3**, 503–516 (1993)
6.12 D.E. Whitney, C.A. Lozinski, J.M. Rourke: Industrial robot forward calibration method and results, ASME J. Dyn. Syst. Meas. Control **108**, 1–8 (1986)
6.13 B.W. Mooring, Z.S. Roth, M.R. Driels: *Fundamentals of Manipulator Calibration* (Wiley, New York 1991)
6.14 K. Lau, R. Hocken, L. Haynes: Robot performance measurements using automatic laser tracking techniques, Robotics Comput. Manuf. **2**, 227–236 (1985)
6.15 C.H. An, C.H. Atkeson, J.M. Hollerbach: *Model-Based Control of a Robot Manipulator* (MIT Press, Cambridge 1988)
6.16 M. Vincze, J.P. Prenninger, H. Gander: A laser tracking system to measure position and orientation of robot end effectors under motion, Int. J. Robotics Res. **13**, 305–314 (1994)
6.17 J.M. McCarthy: *Introduction to Theoretical Kinematics* (MIT Press, Cambridge 1990)
6.18 D.J. Bennet, J.M. Hollerbach: Autonomous calibration of single-loop closed kinematic chains formed by manipulators with passive endpoint constraints, IEEE Trans. Robotics Autom. **7**, 597–606 (1991)
6.19 W.S. Newman, D.W. Osborn: A new method for kinematic parameter calibration via laser line tracking, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (1993) pp. 160–165
6.20 X.-L. Zhong, J.M. Lewis: A new method for autonomous robot calibration, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (1995) pp. 1790–1795
6.21 J.M. Hollerbach, D.M. Lokhorst: Closed-loop kinematic calibration of the RSI 6-DOF hand controller, IEEE Trans. Robotics Autom. **11**, 352–359 (1995)
6.22 A. Nahvi, J.M. Hollerbach, V. Hayward: Closed-loop kinematic calibration of a parallel-drive shoulder joint, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (1994) pp. 407–412
6.23 O. Masory, J. Wang, H. Zhuang: On the accuracy of a Stewart platform – Part II Kinematic calibration and compensation, Proc. IEEE Int. Conf. Robotics Autom., Piscataway (1994) pp. 725–731
6.24 B. Armstrong-Helouvry: *Control of Machines with Friction* (Kluwer, Boston 1991)
6.25 B. Armstrong-Helouvry, P. Dupont, C. de Canudas Wit: A survey of models, analysis tools and compensation methods for the control of machines with friction, Automatica **30**, 1083–1138 (1994)
6.26 F. Aghili, J.M. Hollerbach, M. Buehler: A modular and high-precision motion control system with an integrated motor, IEEE/ASME Trans. Mechatron. **12**, 317–329 (2007)
6.27 W.S. Newman, J.J. Patel: Experiments in torque control of the Adept One robot, Proc. IEEE Int. Conf. Robotics Autom., Piscataway (1991) pp. 1867–1872
6.28 W. Khalil, E. Dombre: *Modeling, Identification and Control of Robots* (Taylor Francis, New York 2002)
6.29 M. Gautier: Dynamic identification of robots with power model, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (1997) pp. 1922–1927
6.30 W. Khalil, M. Gautier, P. Lemoine: Identification of the payload inertial parameters of industrial manipulators, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (2007) pp. 4943–4948
6.31 J. Swevers, W. Verdonck, B. Naumer, S. Pieters, E. Biber: An Experimental Robot Load Identification Method for Industrial Application, Int. J. Robotics Res. **21(8)**, 701–712 (2002)
6.32 P.P. Restrepo, M. Gautier: Calibration of drive chain of robot joints, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (1995) pp. 526–531
6.33 P. Corke: In situ measurement of robot motor electrical constants, Robotica **23**(14), 433–436 (1996)
6.34 S. Van Huffel, J. Vandewalle: *The Total Least Squares Problem: Computational Aspects and Analysis* (SIAM, Philadelphia 1991)
6.35 M. Gautier, S. Briot: Global identification of drive gains parameters of robots using a known pay-

Avidbots Ex1016
Page 183 of 2154

load, ASME J. Dyn. Syst. Meas. Control **136**(5), 051026 (2014)
6.36 W. Khalil, O. Ibrahim: General solution for the dynamic modeling of parallel robots, J. Intell. Robotics Syst. **49**, 19–37 (2007)
6.37 S. Guegan, W. Khalil, P. Lemoine: Identification of the dynamic parameters of the Orthoglide, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (2003) pp. 3272–3277
6.38 K. Schröer: Theory of kinematic modelling and numerical procedures for robot calibration. In: *Robot Calibration*, ed. by R. Bernhardt, S.L. Albright (Chapman Hall, London 1993) pp. 157–196
6.39 H. Zhuang, Z.S. Roth: *Camera-Aided Robot Calibration* (CRC, Boca Raton 1996)
6.40 M. Gautier: Numerical calculation of the base inertial parameters, J. Robotics Syst. **8**, 485–506 (1991)
6.41 J.J. Dongarra, C.B. Mohler, J.R. Bunch, G.W. Stewart: *LINPACK User's Guide* (SIAM, Philadelphia 1979)
6.42 W. Khalil, F. Bennis: Symbolic calculation of the base inertial parameters of closed-loop robots, Int. J. Robotics Res. **14**, 112–128 (1995)
6.43 W. Khalil, S. Guegan: Inverse and direct dynamic modeling of Gough–Stewart robots, IEEE Trans. Robotics Autom. **20**, 754–762 (2004)
6.44 W.H. Press, S.A. Teukolsky, W.T. Vetterling, B.P. Flannery: *Numerical Recipes in C* (Cambridge Univ. Press, Cambridge 1992)
6.45 C.L. Lawson, R.J. Hanson: *Solving Least Squares Problems* (Prentice Hall, Englewood Cliffs 1974)
6.46 P.R. Bevington, D.K. Robinson: *Data Reduction and Error Analysis for the Physical Sciences* (McGraw-Hill, New York 1992)
6.47 B. Armstrong: On finding exciting trajectories for identification experiments involving systems with nonlinear dynamics, Int. J. Robotics Res. **8**, 28–48 (1989)
6.48 J. Fiefer, J. Wolfowitz: Optimum designs in regression problems, Ann. Math. Stat. **30**, 271–294 (1959)
6.49 Y. Sun, J.M. Hollerbach: Observability index selection for robot calibration, Proc. IEEE Int. Conf. Robotics Autom. (ICRA), Piscataway (2008) pp. 831–836
6.50 J.H. Borm, C.H. Menq: Determination of optimal measurement configurations for robot calibration based on observability measure, Int. J. Robotics Res. **10**, 51–63 (1991)
6.51 C.H. Menq, J.H. Borm, J.Z. Lai: Identification and observability measure of a basis set of error parameters in robot calibration, ASME J. Mech. Autom. Des. **111**(4), 513–518 (1989)
6.52 M. Gautier, W. Khalil: Exciting trajectories for inertial parameter identification, Int. J. Robotics Res. **11**, 362–375 (1992)
6.53 M.R. Driels, U.S. Pathre: Significance of observation strategy on the design of robot, J. Robotics Syst. **7**, 197–223 (1990)
6.54 A. Nahvi, J.M. Hollerbach: The noise amplification index for optimal pose selection in robot calibration, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (1996) pp. 647–654
6.55 D. Daney, B. Madeline, Y. Papegay: Choosing measurement poses for robot calibration with local convergence method and Tabu search, Int. J. Robotics Res. **24**(6), 501–518 (2005)
6.56 Y. Sun, J.M. Hollerbach: Active robot calibration algorithm, Proc. IEEE Int. Conf. Robotics Autom. (ICRA), Piscataway (2008) pp. 1276–1281
6.57 T.J. Mitchell: An algorithm for the construction of D-Optimal experimental designs, Technometrics **16**(2), 203–210 (1974)
6.58 J. Swevers, C. Ganseman, D.B. Tukel, J. De Schutter, H. Van Brussel: Optimal robot excitation and identification, IEEE Trans. Robotics Autom. **13**, 730–740 (1997)
6.59 P.O. Vandanjon, M. Gautier, P. Desbats: Identification of robots inertial parameters by means of spectrum analysis, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (1995) pp. 3033–3038
6.60 E. Walter, L. Pronzato: *Identification of Parametric Models from Experimental Data* (Springer, London 1997)
6.61 D.G. Luenberger: *Optimization by Vector Space Methods* (Wiley, New York 1969)
6.62 H.W. Sorenson: Least-squares estimation: from Gauss to Kalman, IEEE Spectr. **7**, 63–68 (1970)
6.63 Z. Roth, B.W. Mooring, B. Ravani: An overview of robot calibration, IEEE J. Robotics Autom. **3**, 377–386 (1987)
6.64 A.E. Bryson Jr., Y.-C. Ho: *Applied Optimal Control* (Hemisphere, Washington 1975)
6.65 M. Gautier, P. Poignet: Extended Kalman fitering and weighted least squares dynamic identification of robot, Control Eng. Pract. **9**, 1361–1372 (2001)
6.66 D.J. Bennet, J.M. Hollerbach, D. Geiger: Autonomous robot calibration for hand-eye coordination, Int. J. Robotics Res. **10**, 550–559 (1991)
6.67 D.J. Bennet, J.M. Hollerbach, P.D. Henri: Kinematic calibration by direct estimation of the Jacobian matrix, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (1992) pp. 351–357
6.68 P.T. Boggs, R.H. Byrd, R.B. Schnabel: A stable and efficient algorithm for nonlinear orthogonal distance regression, SIAM J. Sci. Stat. Comput. **8**, 1052–1078 (1987)
6.69 W.A. Fuller: *Measurement Error Models* (Wiley, New York 1987)
6.70 J.-M. Renders, E. Rossignol, M. Becquet, R. Hanus: Kinematic calibration and geometrical parameter identification for robots, IEEE Trans. Robotics Autom. **7**, 721–732 (1991)
6.71 G. Zak, B. Benhabib, R.G. Fenton, I. Saban: Application of the weighted least squares parameter estimation method for robot calibration, J. Mech. Des. **116**, 890–893 (1994)
6.72 M. Gautier, A. Janot, P.O. Vandanjon: A New Closed-Loop Output Error Method for Parameter Identification of Robot Dynamics, IEEE Trans. Control Syst. Techn. **21**, 428–444 (2013)
6.73 A. Janot, P.O. Vandanjon, M. Gautier: A Generic Instrumental Variable Approach for Industrial Robots Identification, IEEE Trans. Control Syst. Techn. **22**, 132–145 (2014)

Avidbots Ex1016
Page 184 of 2154

Multimedia Contents

7. Motion Planning

Lydia E. Kavraki, Steven M. LaValle

This chapter first provides a formulation of the geometric path planning problem in Sect. 7.2 and then introduces sampling-based planning in Sect. 7.3. Sampling-based planners are general techniques applicable to a wide set of problems and have been successful in dealing with hard planning instances. For specific, often simpler, planning instances, alternative approaches exist and are presented in Sect. 7.4. These approaches provide theoretical guarantees and for simple planning instances they outperform sampling-based planners. Section 7.5 considers problems that involve differential constraints, while Sect. 7.6 overviews several other extensions of the basic problem formulation and proposed solutions. Finally, Sect. 7.8 addresses some important and more advanced topics related to motion planning.

7.1 **Robotics Motion Planning** 139
7.2 **Motion Planning Concepts** 140
7.2.1 Configuration Space 140
7.2.2 Geometric Path Planning Problem . 141
7.2.3 Complexity of Motion Planning 141
7.3 **Sampling-Based Planning** 141
7.3.1 Multi-Query Planners: Mapping the Connectivity of C_{free} ... 142
7.3.2 Single-Query Planners: Incremental Search 143
7.4 **Alternative Approaches** 144
7.4.1 Combinatorial Roadmaps 145
7.4.2 Roadmaps in Higher Dimensions ... 146
7.4.3 Potential Fields 146
7.5 **Differential Constraints** 148
7.5.1 Concepts and Terminology 148
7.5.2 Discretization of Constraints 149
7.5.3 Decoupled Approach 149
7.5.4 Kinodynamic Planning 150
7.6 **Extensions and Variations** 151
7.6.1 Closed Kinematic Chains 151
7.6.2 Manipulation Planning 151
7.6.3 Time-Varying Problems 151
7.6.4 Multiple Robots 152
7.6.5 Uncertainty in Predictability 153
7.6.6 Sensing Uncertainty 153
7.6.7 Optimal Planning 154
7.7 **Advanced Issues** 154
7.7.1 Topology of Configuration Spaces .. 154
7.7.2 Sampling Theory 155
7.7.3 Computational Algebraic Geometry Techniques 156
7.8 **Conclusions and Further Reading** 157
Video-References 158
References 158

7.1 Robotics Motion Planning

A fundamental robotics task is to plan collision-free motions for complex bodies from a start to a goal position among a collection of static obstacles. Although relatively simple, this geometric path planning problem is computationally hard [7.1]. Extensions of this formulation take into account additional problems that are inherited from mechanical and sensor limitations of real robots such as uncertainties, feedback, and differential constraints, which further complicate the development of automated planners. Modern algorithms have been fairly successful in addressing hard instances of the basic geometric problem and

a lot of effort has been devoted to extend their capabilities to more challenging instances. These algorithms have had widespread success in applications beyond robotics, such as computer animation, virtual prototyping, and computational biology. There are many available surveys [7.2–4] and books [7.5–7] that cover modern motion planning techniques and applications.

7.2 Motion Planning Concepts

This section provides a description of the fundamental motion planning problem or the geometric path planning problem. Extensions of this basic formulation to more complicated instances will be discussed later in the chapter and will be revisited throughout this book.

7.2.1 Configuration Space

In path planning, a complete description of the geometry of a robot $\mathcal{A}$ and of a workspace $\mathcal{W}$ is provided. The workspace $\mathcal{W} = \mathbb{R}^N$, in which $N = 2$ or $N = 3$, is a static environment populated with obstacles. The goal is to find a collision-free path for $\mathcal{A}$ to move from an initial position and orientation to a goal position and orientation.

To achieve this, a complete specification of the location of every point on the robot geometry, or a *configuration* $\boldsymbol{q}$, must be provided. The *configuration space*, or C-space ($\boldsymbol{q} \in C$), is the space of all possible configurations. The C-space represents the set of all transformations that can be applied to a robot given its kinematics as described in Chap. 2 (Kinematics). It was recognized early on in motion planning research [7.8, 9] that the C-space is a useful way to abstract planning problems in a unified way. The advantage of this abstraction is that a robot with a complex geometric shape is mapped to a single point in the C-space. The number of degrees of freedom of a robot system is the dimension of the C-space, or the minimum number of parameters needed to specify a configuration.

Let the closed set $\mathcal{O} \subset \mathcal{W}$ represent the *(workspace) obstacle region*, which is usually expressed as a collection of polyhedra, three-dimensional (3-D) triangles, or piecewise-algebraic surfaces. Let the closed set $\mathcal{A}(\boldsymbol{q}) \subset \mathcal{W}$ denote the set of points occupied by the robot when at configuration $\boldsymbol{q} \in C$; this set is usually modeled using the same primitives as used for $\mathcal{O}$. The *C-space obstacle region*, C_{obs}, is defined as

$$C_{\text{obs}} = \{\boldsymbol{q} \in C \mid \mathcal{A}(\boldsymbol{q}) \cap \mathcal{O} \neq \emptyset\}. \tag{7.1}$$

Since $\mathcal{O}$ and $\mathcal{A}(\boldsymbol{q})$ are closed sets in $\mathcal{W}$, the obstacle region is a closed set in C. The set of configurations that avoid collision is $C_{\text{free}} = C \setminus C_{\text{obs}}$, and is called the *free space*.

Simple Examples of C-spaces

Translating Planar Rigid Bodies. The robot's configuration can be specified by a reference point (x, y) on the planar rigid body relative to some fixed coordinate frame. Therefore the C-space is equivalent to $\mathbb{R}^2$. Figure 7.1 gives an example of a C-space for a triangular robot and a single polygonal obstacle. The obstacle region in the C-space can be traced by sliding the robot around the workspace obstacle to find the constraints on all $\boldsymbol{q} \in C$. Motion planning for the robot is now equivalent to motion planning for a point in the C-space.

Planar Arms. Figure 7.2 gives an example of a two-joint planar arm. The bases of both links are pinned, so that they can only rotate around the joints, and there are no joint limits. For this arm, specifying the rotational parameters θ_1 and θ_2 provides the configuration. Each joint angle θ_i corresponds to a point on the unit circle $\mathbb{S}^1$ and the C-space is $\mathbb{S}^1 \times \mathbb{S}^1 = T^2$, the two-dimensional (2-D) torus shown in Fig. 7.2. For a higher number of

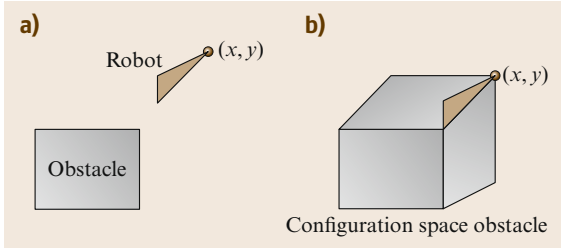


Fig.7.1a,b A robot translating in the plane: **(a)** a triangular robot moves in a workspace with a single rectangular obstacle. **(b)** The C-space obstacle

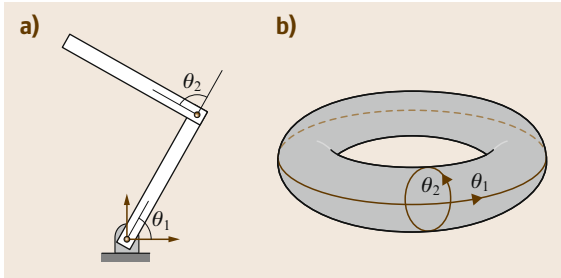


Fig. 7.2 **(a)** A two-joint planar arm in which the links are pinned and there are no joint limits. **(b)** The C-space

Avidbots Ex1016
Page 186 of 2154

links without joint limits, the C-space can be similarly defined as

$$\mathcal{C} = \mathbb{S}^1 \times \mathbb{S}^1 \times \cdots \times \mathbb{S}^1 \,. \tag{7.2}$$

If a joint has limits, then each corresponding $\mathbb{S}^1$ is often replaced with $\mathbb{R}$, even though it is a finite interval. If the base of the planar arm is mobile and not pinned, then the additional translation parameters must also be considered in the arm's configuration

$$\mathcal{C} = \mathbb{R}^2 \times \mathbb{S}^1 \times \mathbb{S}^1 \times \cdots \times \mathbb{S}^1 \,. \tag{7.3}$$

Additional examples of C-spaces are provided in Sect. 7.7.1, where topological properties of configuration spaces are discussed.

7.2.2 Geometric Path Planning Problem

The basic motion planning problem, also known as the *piano mover's problem* [7.1], is defined as follows.
Given:

1. A *workspace* $\mathcal{W}$, where either $\mathcal{W} = \mathbb{R}^2$ or $\mathcal{W} = \mathbb{R}^3$.
2. An *obstacle region* $\mathcal{O} \subset \mathcal{W}$.
3. A *robot* defined in $\mathcal{W}$. Either a rigid body $\mathcal{A}$ or a collection of m links: $\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_m$.
4. The *configuration space* $\mathcal{C}$ ($\mathcal{C}_{\text{obs}}$ and $\mathcal{C}_{\text{free}}$ are then defined).
5. An *initial configuration* $\boldsymbol{q}_{\text{I}} \in \mathcal{C}_{\text{free}}$.
6. A *goal configuration* $\boldsymbol{q}_{\text{G}} \in \mathcal{C}_{\text{free}}$. The initial and goal configuration are often called a *query* $(\boldsymbol{q}_{\text{I}}, \boldsymbol{q}_{\text{G}})$.

Compute a (continuous) path, $\tau : [0, 1] \to \mathcal{C}_{\text{free}}$, *such that* $\tau(0) = \boldsymbol{q}_{\text{I}}$ *and* $\tau(1) = \boldsymbol{q}_{\text{G}}$.

7.2.3 Complexity of Motion Planning

The main complications in motion planning are that it is not easy to directly compute $\mathcal{C}_{\text{obs}}$ and $\mathcal{C}_{\text{free}}$, and the dimensionality of the C-space is often quite high. In terms of computational complexity, the piano mover's problem was studied early on and it was shown to be PSPACE-hard by *Reif* [7.1]. A series of polynomial-time algorithms for problems with fixed dimension suggested an exponential dependence on the problem dimensionality [7.10, 11]. A single exponential-time algorithm in the C-space dimensionality was proposed by *Canny* and showed that the problem is PSPACE-complete [7.12]. Although impractical, the algorithm serves as an upper bound on the general version of the basic motion planning problem. It applies computational algebraic geometry techniques for modeling the C-space in order to construct a *roadmap*, a one-dimensional (1-D) subspace that captures the connectivity of $\mathcal{C}_{\text{free}}$. Additional details about such techniques can be found in Sect. 7.7.3.

The complexity of the problem motivated work in path planning research. One direction was to study subclasses of the general problem for which polynomial time algorithms exist [7.13]. Even some simpler, special cases of motion planning, however, are at least as challenging, for example, the case of a finite number of translating, axis-aligned rectangles in $\mathbb{R}^2$ is PSPACE-hard as well [7.14]. Some extensions of motion planning are even harder. For example, a certain form of planning under uncertainty in 3-D polyhedral environment is NEXPTIME-hard [7.15]. The hardest problems in NEXPTIME are believed to require doubly exponential time to solve.

A different direction was the development of alternative planning paradigms that were practical under realistic assumptions. Many combinatorial approaches can efficiently construct 1-D roadmaps for specific 2-D or 3-D problems. Potential field-based approaches define vector fields which can be followed by a robot towards the goal. Both approaches, however, do not scale well in the general case. They will be described in Sect. 7.4. An alternative paradigm, sampling-based planning, is a general approach that has been shown to be successful in practice for many challenging problems. It avoids the exact geometric modeling of the C-space but it cannot provide the guarantees of a complete algorithm. Complete and exact algorithms are able to detect that no path can be found. Instead sampling-based planning offers a lower level of completeness guarantee. This paradigm is described in the following section.

7.3 Sampling-Based Planning

Sampling-based planners are described first because they are the method of choice for a very general class of problems. The following section will describe other planners, some of which were developed before the sampling-based framework. The key idea in sampling-based planning is to exploit advances in collision detection algorithms that compute whether a single configuration is collision free. Given this simple primitive, a planner samples different configurations to construct a data structure that stores 1-D C-space curves, which represent collision-free paths. In this way, sampling-based planners do not access the C-space obstacles directly but only through the collision detector and the constructed data structure. Using this level of abstrac-

Avidbots Ex1016
Page 187 of 2154

tion, the planners are applicable to a wide range of problems by tailoring the collision detector to specific robots and applications.

A standard for sampling-based planners is to provide a weaker, but still interesting, form of completeness: *if a solution path exists, the planner will eventually find it*. Giving up on the stronger form of completeness, which also requires failure to be reported in finite time, these techniques are able to solve in practice problems with more than three degrees of freedom that complete approaches cannot address. More details on this weaker form of completeness are provided in Sect. 7.7.2.

Different planners follow different approaches on how to sample configurations and what kind of data structures they construct. Section 7.7.2 provides a deeper insight on sampling issues. A typical classification of sampling-based planners is between multi-query and single-query approaches:

- In the first category, the planners construct a roadmap, an undirected graph G that is precomputed once so as to *map the connectivity properties of* C_{free}. After this step, multiple queries in the same environment can be answered using only the constructed roadmap. Such planners are described in Sect. 7.3.1.
- Planners in the second category build tree data structures on the fly given a planning query. They attempt to focus on exploring the part of the C-space that will lead to solving a specific query as fast as possible. They are described in Sect. 7.3.2.

Both approaches, however, make similar use of a collision checking primitive. The objective of a collision detector is to report all geometric contacts between objects given their geometries and transformations [7.16–18]. The availability of packages that were able to answer collision queries in a fraction of a second was critical to the development of sampling-based planners. Modern planners use collision detectors as a *black box*. Initially the planner provides the geometries of all the involved objects and specifies which of them are mobile. Then, in order to validate a robot configuration, a planner provides the corresponding robot transformation and a collision detector responds on whether the objects collide or not. Many packages represent the geometric models hierarchically, avoid computing all-pairwise interactions, and conduct a binary search to evaluate collisions. Except from configurations, a planner must also validate entire paths. Some collision detectors return distance-from-collision information, which can be used to infer that entire neighborhoods in C are valid. It is often more expensive, however, to extract this information; instead paths are usually validated point by point using a small stepping size either incrementally or by employing binary search. Some collision detectors are incremental by design, which means that they can be faster by reusing information from a previous query [7.16]. Examples of problems solved by sampling-based planners are shown in VIDEO 24 and VIDEO 17.

7.3.1 Multi-Query Planners: Mapping the Connectivity of C_{free}

Planners that aim to answer multiple queries for a certain static environment use a preprocessing phase during which they attempt to map the connectivity properties of C_{free} onto a roadmap. This roadmap has the form of a graph G, with vertices as configurations and edges as paths. A union of 1-D curves is a roadmap G if it satisfies the following properties:

1. *Accessibility*: From any $\boldsymbol{q} \in C_{\text{free}}$, it is simple and efficient to compute a path $\tau : [0,1] \to C_{\text{free}}$ such that $\tau(0) = \boldsymbol{q}$ and $\tau(1) = \boldsymbol{s}$, in which $\boldsymbol{s}$ may be any point in $S(G)$. $S(G)$ is the *swath* of G, the union of all configurations reached by all edges and vertices. This means that it is always possible to connect a planning query pair $\boldsymbol{q}_{\text{I}}$ and $\boldsymbol{q}_{\text{G}}$ to some $\boldsymbol{s}_{\text{I}}$ and $\boldsymbol{s}_{\text{G}}$, respectively, in $S(G)$.
2. *Connectivity preserving*: The second condition requires that, if there exists a path $\tau : [0,1] \to C_{\text{free}}$ such that $\tau(0) = \boldsymbol{q}_{\text{I}}$ and $\tau(1) = \boldsymbol{q}_{\text{G}}$, then there also exists a path $\tau' : [0,1] \to S(G)$, such that $\tau'(0) = \boldsymbol{s}_{\text{I}}$ and $\tau'(1) = \boldsymbol{s}_{\text{G}}$. Thus, solutions are not missed because G fails to capture the connectivity of C_{free}.

The probabilistic roadmap method (PRM) approach [7.19] attempts to approximate such a roadmap G in a computationally efficient way. The preprocessing phase of PRM, which can be extended to sampling-based roadmaps in general, follows these steps:

1. *Initialization*: Let $G(V, E)$ represent an undirected *graph*, which is initially empty. Vertices of G will correspond to collision-free configurations, and edges to collision-free paths that connect vertices.
2. *Configuration sampling*: A configuration $\alpha(i)$ is sampled from C_{free} and added to the vertex set V. $\alpha(\cdot)$ is an infinite, dense sample sequence and $\alpha(i)$ is the i-th point in that sequence.
3. *Neighborhood computation*: Usually, a *metric* is defined in the C-space, $\rho : C \times C \to \mathbb{R}$. Vertices $\boldsymbol{q}$ already in V are then selected as part of $\alpha(i)$'s neighborhood if they have small distance according to ρ.
4. *Edge consideration*: For those vertices $\boldsymbol{q}$ that do not belong in the same connected component of G with

Avidbots Ex1016
Page 188 of 2154

$\alpha(i)$ the algorithm attempts to connect them with an edge.

5. *Local planning method*: Given $\alpha(i)$ and $\boldsymbol{q} \in C_{\text{free}}$ a module is used that attempts to construct a path $\tau_s : [0,1] \to C_{\text{free}}$ such that $\tau(0) = \alpha(i)$ and $\tau(1) = \boldsymbol{q}$. Using collision detection, τ_s must be checked to ensure that it does not cause a collision.
6. *Edge insertion*: Insert τ_s into E, as an edge from $\alpha(i)$ to $\boldsymbol{q}$.
7. *Termination*: The algorithm is typically terminated when a predefined number of collision-free vertices N has been added in the roadmap.

The algorithm is incremental in nature. Computation can be repeated by starting from an already existing graph. A general sampling-based roadmap is summarized in Algorithm 7.1.

Algorithm 7.1 Sampling-Based Roadmap

N: number of nodes to include in the roadmap

```
1:  G.init(); i ← 0;
2:  while i < N do
3:    if α(i) ∈ C_free then
4:      G.add_ vertex(α(i)); i ← i + 1;
5:      for q ∈ NEIGHBORHOOD(α(i),G) do
6:        if CONNECT (α(i), q) then
7:          G.add_ edge (α(i), q);
8:        end if
9:      end for
10:   end if
11: end while
```

An illustration of the algorithm's behavior is depicted in Fig. 7.3. To solve a query, $\boldsymbol{q}_{\text{I}}$ and $\boldsymbol{q}_{\text{G}}$ are connected to the roadmap, and graph search is performed.

For the original PRM [7.19], the configuration $\alpha(i)$ was produced using random sampling. For the connection step between $\boldsymbol{q}$ and $\alpha(i)$, the algorithm used straight line paths in the C-space. In some cases a connection was attempted if $\boldsymbol{q}$ and $\alpha(i)$ were in the same connected component in order to improve path quality. There have been many subsequent works that try to improve the roadmap quality while using fewer samples. Methods for concentrating samples at or near the boundary of C_{free} are presented in [7.20, 21]. Methods that move samples as far from the boundary as possible appear in [7.22, 23]. Deterministic sampling techniques, including grids, appear in [7.24]. A method of pruning vertices based on mutual visibility that leads to a dramatic reduction in the number of roadmap vertices appears in [7.25]. Theoretical analysis of sampling-based roadmaps appears in [7.24, 26, 27] and is briefly discussed in Sect. 7.7.2. An experimental comparison of sampling-based roadmap variants appears in [7.28]. One difficulty in these roadmap approaches is identifying narrow passages. One proposal is to use the *bridge test* for identifying these [7.29]. For other PRM-based works, see [7.30–34]. Extended discussion of the topic can be found in [7.5, 7].

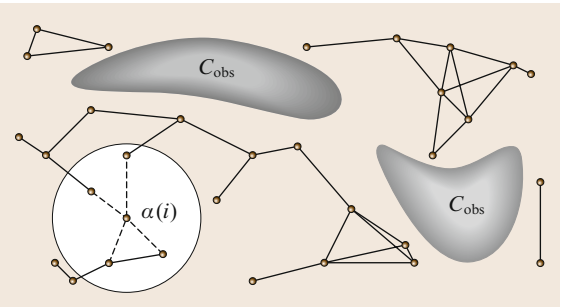


Fig. 7.3 The sampling-based roadmap is constructed incrementally by attempting to connect each new sample, $\alpha(i)$, to nearby vertices in the roadmap

7.3.2 Single-Query Planners: Incremental Search

Single-query planning methods focus on a single initial–goal configuration pair. They probe and search the continuous C-space by extending tree data structures initialized at these known configurations and eventually connecting them. Most single-query methods conform to the following template:

1. *Initialization*: Let $G(V, E)$ represent an undirected *search graph*, for which the vertex set V contains a vertex for one (usually $\boldsymbol{q}_{\text{I}}$) or more configurations in C_{free}, and the edge set E is empty. Vertices of G are collision-free configurations, and edges are collision-free paths that connect vertices.
2. *Vertex selection method*: Choose a vertex $\boldsymbol{q}_{\text{cur}} \in V$ for expansion.
3. *Local planning method*: For some $\boldsymbol{q}_{\text{new}} \in C_{\text{free}}$, which may correspond to an existing vertex in V but on a different tree or a sampled configuration, attempt to construct a path $\tau_s : [0,1] \to C_{\text{free}}$ such that $\tau(0) = \boldsymbol{q}_{\text{cur}}$ and $\tau(1) = \boldsymbol{q}_{\text{new}}$. Using collision detection, τ_s must be checked to ensure that it does not cause a collision. If this step fails to produce a collision-free path segment, then go to Step 2.
4. *Insert an edge in the graph*: Insert τ_s into E, as an edge from $\boldsymbol{q}_{\text{cur}}$ to $\boldsymbol{q}_{\text{new}}$. If $\boldsymbol{q}_{\text{new}}$ is not already in V, then it is inserted.
5. *Check for a solution*: Determine whether G encodes a solution path.

Avidbots Ex1016
Page 189 of 2154

6. *Return to Step 2*: Iterate unless a solution has been found or some termination condition is satisfied, in which case the algorithm reports failure.

During execution, G may be organized into one or more trees. This leads to:

1. *Unidirectional* methods, which involve a single tree, usually rooted at $\boldsymbol{q}_\mathrm{I}$ [7.35],
2. *Bidirectional* methods, which involve two trees, typically rooted at $\boldsymbol{q}_\mathrm{I}$ and $\boldsymbol{q}_\mathrm{G}$ [7.35], and
3. *Multidirectional* methods, which may have more than two trees [7.36, 37].

The motivation for using more than one tree is that a single tree may become trapped trying to find an exit through a narrow opening. Traveling in the opposite direction, however, may be easier. As more trees are considered it becomes more complicated to determine which connections should be made between trees.

Rapidly Exploring Dense Trees

The important idea with this family of techniques is that the algorithm must incrementally explore the properties of the C-space. An algorithm that achieves this objective is the rapidly exploring random tree (RRT) [7.35], which can be generalized to the rapidly exploring dense tree (RDT) for any dense sampling, deterministic or random [7.7]. The basic idea is to induce a *Voronoi bias* in the exploration process by selecting for expansion the point in the tree that is closest to $\alpha(i)$ in each iteration. Using random samples, the probability that a vertex is chosen is proportional to the volume of its Voronoi region. The tree construction is outlined as:

Algorithm 7.2 Rapidly Exploring Dense Trees

k: the exploration steps of the algorithm

```
1: G.init(q_I);
2: for i = 1 to k do
3:     G.add_vertex(α(i));
4:     q_n ← NEAREST(S(G), α(i));
5:     G.add_edge(q_n, α(i));
6: end for
```

The tree starts at $\boldsymbol{q}_\mathrm{I}$, and in each iteration, an edge and vertex are added (Fig. 7.4).

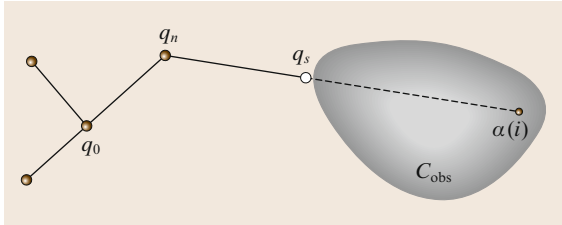


Fig. 7.4 If there is an obstacle, the edge travels up to the obstacle boundary, as far as allowed by the collision detection algorithm

So far, the problem of reaching $\boldsymbol{q}_\mathrm{G}$ has not been explained. There are several ways to use RDTs in a planning algorithm. One approach is to bias $\alpha(i)$ so that $\boldsymbol{q}_\mathrm{G}$ is frequently chosen (perhaps once every 50 iterations). A more efficient approach is to develop a bidirectional search by growing two trees, one from each of $\boldsymbol{q}_\mathrm{I}$ and $\boldsymbol{q}_\mathrm{G}$. Roughly half of the time is spent expanding each tree in the usual way, while the other half is spend attempting to connect the trees. The simplest way to connect trees is to let the newest vertex of one tree be a substitute for $\alpha(i)$ in extending the other. This tricks one RDT into attempting to connect to the other while using the basic expansion algorithm [7.38]. Several works have extended, adapted, or applied RDTs in various applications [7.37, 39–42]. Detailed descriptions can be found in [7.5, 7].

Other Tree Algorithms

Planners based on the idea of expansive spaces are presented in [7.43–45]. In this case, the algorithm forces exploration by choosing vertices for expansion that have fewer points in a neighborhood around them. In [7.46], additional performance is obtained by self-tuning random walks, which focus virtually all of their effort on exploration. Other successful tree-based algorithms include the path-directed subdivision tree algorithm [7.47] and some of its variants [7.48]. In the literature, it is sometimes hard to locate tree-based planners for ordinary path planning problems as many of them (including RRT) were designed and/or applied to more complex problems (Sect. 7.5.4). Their performance is nevertheless excellent for a variety of path planing problems.

7.4 Alternative Approaches

Alternative approaches to the sampling-based paradigm include potential-field-based techniques and combinatorial methods that also produce roadmaps, such as cell decompositions. These algorithms are able to elegantly and efficiently solve a narrow class of problems, and are much preferred over the algorithms of Sect. 7.3 in

Avidbots Ex1016
Page 190 of 2154

these cases. Most of the combinatorial algorithms are of theoretical interest, whereas sampling-based algorithms are motivated primarily by performance issues in challenging applications. Nevertheless, given some abstractions, the combinatorial algorithms can be used to solve practical problems such as autonomous navigation of mobile planar robots.

7.4.1 Combinatorial Roadmaps

Several algorithms exist for the case in which $C = \mathbb{R}^2$ and C_{obs} is polygonal. Most of these cannot be directly extended to higher dimensions; however, some of the general principles remain the same. The *maximum clearance roadmap* (or *retraction method* [7.49]) constructs a roadmap that keeps paths as far from the obstacles as possible. Paths are contributed to the roadmap from the three cases shown in Fig. 7.5, which correspond to all ways to pair together polygon features. The roadmap can be made naively in time $O(n^4)$ by generating all curves shown in Fig. 7.5 for all possible pairs, computing their intersections, and tracing out the roadmap. Several algorithms exist that provide better asymptotic running time [7.50], but they are considerably more difficult to implement. The best-known algorithm runs in $O(n \lg n)$ time in which n is the number of roadmap curves [7.51].

An alternative is to compute a *shortest-path roadmap* [7.52], as shown in Fig. 7.6. This is different than the roadmap presented in the previous section because paths may actually touch the obstacles, which must be allowed for paths to be optimal. The roadmap vertices are the reflex vertices of C_{obs}, which are vertices for which the interior angle is greater than π. An edge exists in the roadmap if and only if a pair of vertices is mutually visible and the line through them pokes into C_{free} when extended outward from each vertex (such lines are called *bitangents*). An $O(n^2 \lg n)$-time construction algorithm can be formed by using a radial sweep algorithm from each reflex vertex. It can theoretically be computed in time $O(n^2 + m)$, in which m is the total number of edges in the roadmap [7.53].

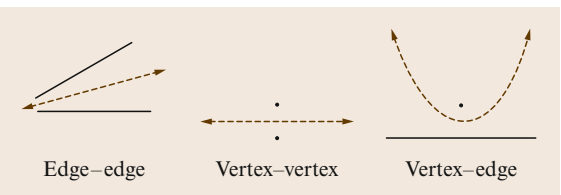


Fig. 7.5 Voronoi roadmap pieces are generated in one of three possible cases. The third case leads to a quadratic curve

Figure 7.7 illustrates the *vertical cell decomposition* approach. The idea is to decompose C_{free} into cells that are trapezoids or triangles. Planning in each cell is trivial because it is convex. A roadmap is made by placing a point in the center of each cell and each boundary between cells. Any graph search algorithm can be used to find a collision-free path quickly. The cell decomposition can be constructed in $O(n \lg n)$ time using the *plane-sweep principle* [7.54, 55]. Imagine that a vertical line sweeps from $x = -\infty$ to $x = \infty$, stopping at places where a polygon vertex is encountered. In these cases, a cell boundary may be necessary above and/or below the vertex. The order in which segments stab the vertical line is maintained in a balanced search tree. This enables the determination of the vertical cell boundary limits in time $O(\lg n)$. The whole algorithm runs in time $O(n \lg n)$ because there are $O(n)$ vertices at which the sweep line can stop (also, the vertices need to be sorted at the outset, which requires time $O(n \lg n)$).

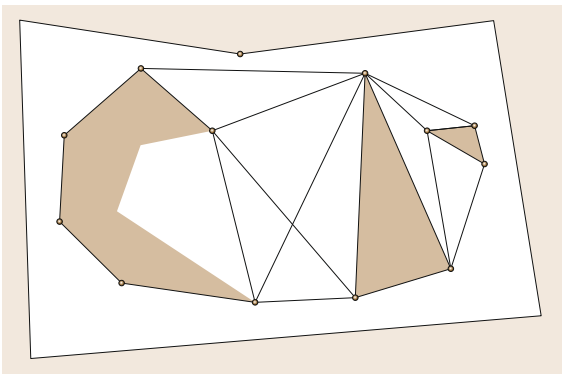


Fig. 7.6 The shortest-path roadmap includes edges between consecutive reflex vertices on C_{obs} and also bitangent edges

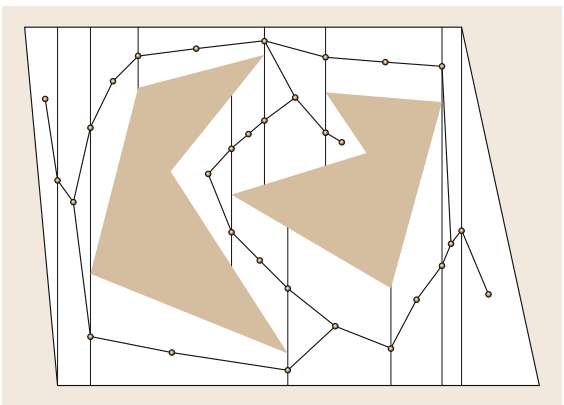


Fig. 7.7 The roadmap derived from the vertical cell decomposition

Avidbots Ex1016
Page 191 of 2154

7.4.2 Roadmaps in Higher Dimensions

It would be convenient if the methods of Sect. 7.4.1 directly extend into higher dimensions. Although this unfortunately does not occur, some of the general ideas extend. To consider a cell decomposition in higher dimensions, there are two main requirements: (1) each cell should be simple enough that motion planning within a cell is trivial; (2) the cells should fit together nicely. A sufficient condition for the first requirement is that cells are convex; more general shapes may be allowed; however, the cells should not contain holes under any circumstances. For the second requirement, a sufficient condition is that the cells can be organized into a *singular complex*. This means that for any two d-dimensional cells for $d \leq n$, if the boundaries of the cells intersect, then the common boundary must itself be a complete cell (of lower dimension).

In two-dimensional polygonal C-spaces, triangulation methods define nice cell decompositions that are appropriate for motion planning. Finding good triangulations, which for example means trying to avoid thin triangles, is given considerable attention in computational geometry [7.55]. Determining the decomposition of a polygonal obstacle region with holes that uses the smallest number of convex cells is *NP*-hard [7.56]. Therefore, we are willing to tolerate nonoptimal decompositions.

In three-dimensional C-spaces, if C_{obs} is polyhedral, then the vertical decomposition method directly extends by applying the plane sweep recursively, for example, the critical events may occur at each z coordinate, at which point changes a 2-D vertical decomposition over the x and y coordinates are maintained. The polyhedral case is obtained for a translating polyhedral robot among polyhedral obstacles in $\mathbb{R}^3$; however, for most interesting problems, C_{obs} becomes nonlinear. Suppose $C = \mathbb{R}^2 \times \mathbb{S}^1$, which corresponds to a robot that can translate and rotate in the plane. Suppose the robot and obstacles are polygonal. For the case of a line-segment robot, an $O(n^5)$ algorithm that is not too difficult to implement is given in [7.57]. The approaches for more general models and C-spaces are extremely difficult to use in practice; they are mainly of theoretical interest and are summarized in Sect. 7.7.3.

7.4.3 Potential Fields

A different approach for motion planning is inspired from obstacle avoidance techniques [7.58]. It does not explicitly construct a roadmap, but instead constructs a differentiable real-valued function $U : \mathbb{R}^m \to \mathbb{R}$, called a potential function, that guides the motion of the moving object. The potential is typically constructed so that it consists of an attractive component $U_{\text{a}}(\boldsymbol{q})$, which pulls the robot towards the goal, and a repulsive component $U_{\text{r}}(\boldsymbol{q})$, which pushes the robot away from the obstacles, as shown in Fig. 7.8. The gradient of the potential function is the vector $\nabla U(\boldsymbol{q}) = DU(\boldsymbol{q})^{\text{T}} = \left[\frac{\partial U}{\partial q_1}(\boldsymbol{q}), \ldots, \frac{\partial U}{\partial q_m}(\boldsymbol{q})\right]^{\text{T}}$, which

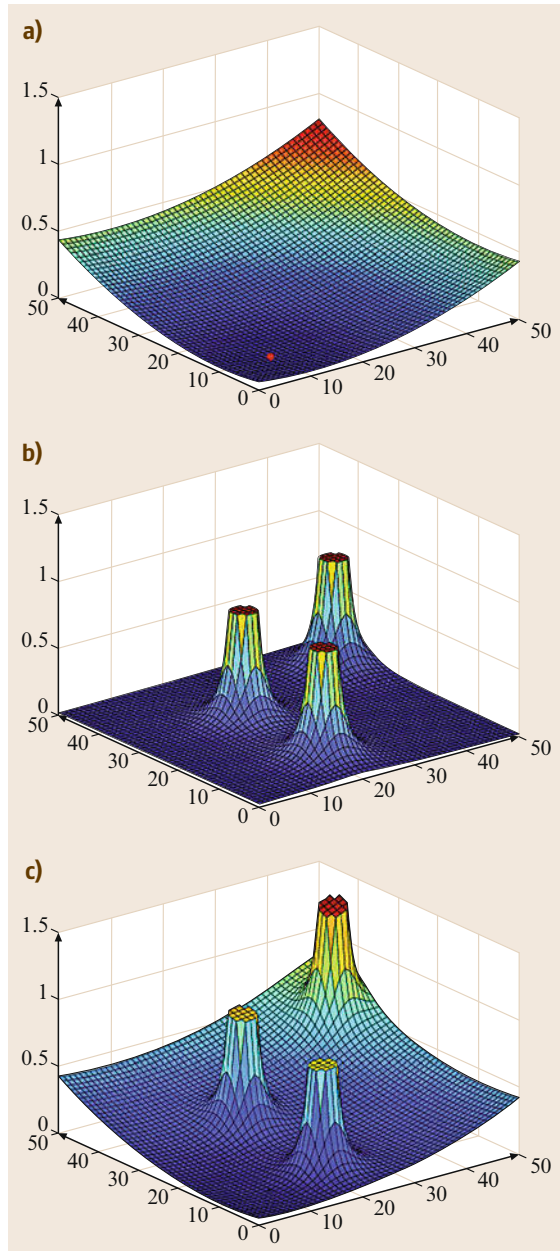


Fig.7.8a–c An attractive and a repulsive component define a potential function. **(a)** An attractive potential, **(b)** a repulsive potential, **(c)** an attractive and repulsive component define a potential function

Avidbots Ex1016
Page 192 of 2154

points in the direction that locally maximally increases U. After the definition of U, a path can be computed by starting from $\boldsymbol{q}_\mathrm{I}$ and applying *gradient descent*:

1. $\boldsymbol{q}(0) = \boldsymbol{q}_\mathrm{I}$; $i = 0$;
2. **while** $\nabla U(\boldsymbol{q}(i)) \neq 0$ **do**
3. $\quad \boldsymbol{q}(i+1) = \boldsymbol{q}(i) + \nabla U(\boldsymbol{q}(i))$
4. $\quad i = i + 1$

However, this gradient-descent approach does not guarantee a solution to the problem. Gradient descent can only reach a local minimum of $U(\boldsymbol{q})$, which may not correspond to the goal state $\boldsymbol{q}_\mathrm{G}$, as shown in Fig. 7.9.

A planner that makes uses of potential functions and attempts to avoid the issue of local minima is the *randomized potential planner* [7.59]. The idea is to combine potential functions with random walks by employing multiple planning modes. In one mode, gradient descent is applied until a local minimum is reached. Another mode uses random walks to try to escape local minima. A third mode performs backtracking whenever several attempts to escape a local minimum have failed. In many ways, this approach can be considered as a sampling-based planner. It also provides the weaker completeness guarantee but it requires parameter tuning. Recent sampling-based methods achieve better performance by spending more time exploring the space, rather than focusing heavily on a potential function.

The gradient of the potential function can be also used to define a *vector field*, which assigns a motion for the robot at any arbitrary configuration $\boldsymbol{q} \in C$. This is an important advantage of the approach, beyond its computational efficiency, since it does not only compute a single path, but also a *feedback control* strategy. This makes the approach more robust against control and sensing errors. Most of the techniques in feedback motion planning are based on the idea of *navigation functions* [7.60], which are potential functions properly constructed so as to have a single minimum. A function $\phi : C_\mathrm{free} \to [0, 1]$ is called a navigation function if it:

- Is smooth (or at least C^k for $k \geq 2$),
- Has a unique minimum at $\boldsymbol{q}_\mathrm{G}$ in the connected component of the free space that contains $\boldsymbol{q}_\mathrm{G}$,
- Is uniformly maximal on the free-space boundaries,
- and is Morse, which means that all its critical points, such as saddle points, are isolated and can be avoided with small random perturbations.

Navigation functions can be constructed for sphere boundary spaces centered at $\boldsymbol{q}_\mathrm{I}$ that contain only spherical obstacles, as illustrated in Fig. 7.10. Then they can be extended to a large family of C-spaces that are diffeomorphic to sphere spaces, such as star-shaped spaces, as shown in Fig. 7.10. A more elaborate description of strategies for feedback motion planning will be presented in Chap. 47.

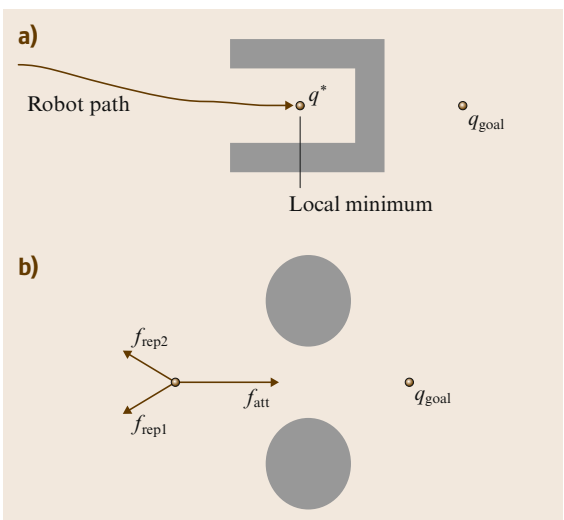


Fig.7.9a,b Two examples of the local minimum problem with potential functions

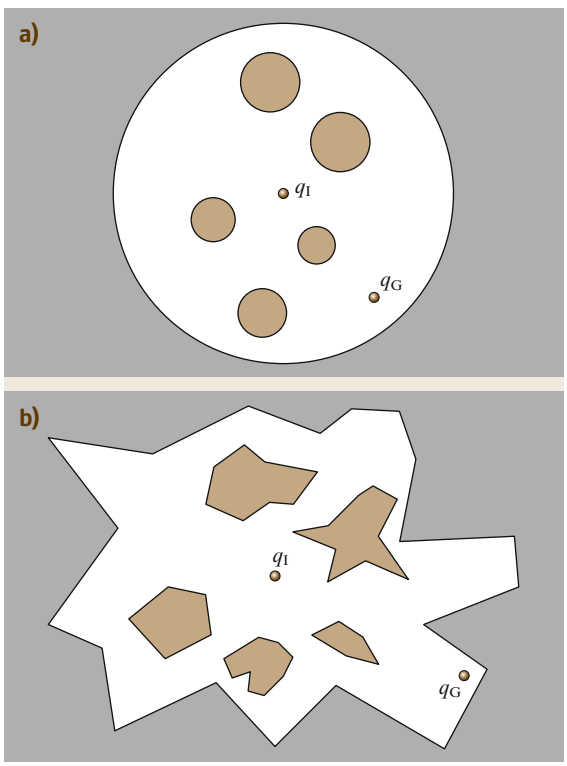


Fig.7.10a,b Examples of (a) sphere and (b) star spaces

Avidbots Ex1016
Page 193 of 2154

Putting the issue of local minima aside, another major challenge for such potential function based approaches is constructing and representing the C-space in the first place. This issue makes the applications of these techniques too complicated for high-dimensional problems.

7.5 Differential Constraints

Robot motions must usually conform to both global and local constraints. Global constraints on C have been considered in the form of obstacles and possibly joint limits. Local constraints are modeled with differential equations, and are therefore called *differential constraints*. These limit the velocities, and possibly accelerations, at every point due to kinematic considerations, such as wheels in contact, and dynamical considerations, such as the conservation of angular momentum.

7.5.1 Concepts and Terminology

Let $\dot{\boldsymbol{q}}$ denote a velocity vector. Differential constraints on C can be expressed either *implicitly* in the form $g_i(\boldsymbol{q}, \dot{\boldsymbol{q}}) = 0$ or *parametrically* in the form $\dot{\boldsymbol{x}} = f(\boldsymbol{q}, \boldsymbol{u})$. The implicit form is more general but often more difficult to understand and utilize. In the parametric form, a vector-valued equation indicates the velocity that is obtained for a given $\boldsymbol{q}$ and $\boldsymbol{u}$, in which $\boldsymbol{u}$ is an *input*, chosen from some *input space*, U. Let T denote an interval of time, starting at $t = 0$.

To model dynamics, the concepts are extended into a *phase space* X of the C-space. Usually each point $\boldsymbol{x} \in X$ represents both a configuration and velocity, $\boldsymbol{x} = (\boldsymbol{q}, \dot{\boldsymbol{q}})$. Both implicit and parametric representations are possible, yielding $g_i(\boldsymbol{x}, \dot{\boldsymbol{x}}) = 0$ and $\dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u})$, respectively. The latter is a common *control system* definition. Note that $\dot{\boldsymbol{x}} = (\dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}})$, which implies that acceleration constraints and full system dynamics can be expressed.

Planning in the state space X could lead to a straightforward definition of X_{obs} by declaring $\boldsymbol{x} \in X_{\text{obs}}$ if and only if $\boldsymbol{q} \in C_{\text{obs}}$ for $\boldsymbol{x} = (\boldsymbol{q}, \dot{\boldsymbol{q}})$. However, another interesting possibility exists which provides some intuition about the difficulty of planning with dynamics. This possibility is based on the notion of a *region of inevitable collision*, which is defined as

$$X_{\text{ric}} = \{\boldsymbol{x}(0) \in X \mid \text{for any } \tilde{u} \in \mathcal{U}_\infty, \exists t > 0 \text{ such that } \boldsymbol{x}(t) \in X_{\text{obs}}\} , \tag{7.4}$$

in which $\boldsymbol{x}(t)$ is the state at time t obtained by integrating the control function $\tilde{u} : T \to U$ from $\boldsymbol{x}(0)$. The set $\mathcal{U}_\infty$ is a predefined set of all possible control functions. X_{ric} denotes the set of states in which the robot is either in collision or, because of momentum, it cannot do anything to avoid collision. It can be considered as an invisible obstacle region that grows with speed (Fig. 7.11).

Under the general heading of *planning under differential constraints*, there are many important categories of problems that have received considerable attention in research literature. The term *nonholonomic planning* was introduced for wheeled mobile robots [7.61]. A simple example is that a car cannot move sideways, thereby making parallel parking more difficult. In general, a *nonholonomic constraint* is a differential equality constraint that cannot be integrated into a constraint that involves no derivatives. Typically, nonholonomic constraints that appear in robotics are *kinematic*, and arise from wheels in contact [7.62]. Nonholonomic constraints may also arise from dynamics.

If a planning problem involves constraints on at least velocity and acceleration, the problem is often referred to as *kinodynamic planning* [7.63]. Usually, the model expresses a *fully actuated system*, which means that it can be expressed as $\ddot{\boldsymbol{q}} = h(\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{u})$, in which U contains an open set that includes the origin of $\mathbb{R}^n$ (here, n is the dimension of both U and C). It is possible for a problem to be nonholonomic, kinodynamic, both, or neither; however, in recent times, the terms are not used with much precision.

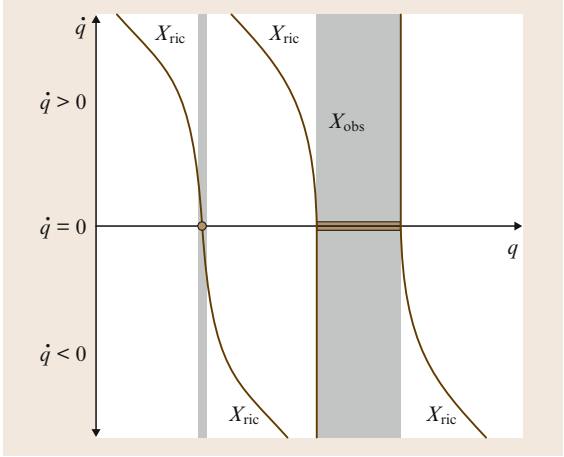


Fig. 7.11 The region of inevitable collision grows quadratically with the speed

Avidbots Ex1016
Page 194 of 2154

Trajectory planning is another important term, which has referred mainly to the problem of determining both a path and velocity function for a robot arm (e.g., PUMA 560). In the treatment below, all of these will be referred to as *planning under differential constraints.*

7.5.2 Discretization of Constraints

The only known methods for complete and optimal planning under differential constraints in the presence of obstacles are for the double integrator system with $X = \mathbb{R}$ [7.64] and $X = \mathbb{R}^2$ [7.65]. To develop planning algorithms in this context, several discretizations are often needed. For ordinary motion planning, only C needed to be discretized; with differential constraints, T and possibly also U require discretization, in addition to C (or X).

Discretization of the differential constraints is one of the most important issues. To solve challenging planning problems efficiently, it is often necessary to define *motion primitives* for the particular dynamical system [7.40, 66, 67]. One of the simplest ways to discretize the differential constraints is to construct a *discrete-time model*, which is characterized by three aspects:

1. The time interval T is partitioned into intervals of length Δt. This enables stages to be assigned, in which stage k indicates that $(k-1)\Delta t$ time has elapsed.
2. A finite subset U_{d} of the action space U is chosen. If U is already finite, then this selection may be $U_{\mathrm{d}} = U$.
3. The action $\boldsymbol{u}(t)$ must remain constant over each time interval.

From an initial state, $\boldsymbol{x}$, a *reachability tree* can be formed by applying all sequences of discretized actions. Figure 7.12 shows the path of this tree for the *Dubins car*, which is a kinematic model of a car that drives in the plane at unit speed and cannot move in reverse. The edges of the tree are circular arcs and line segments. For general systems, each trajectory segment in the tree is determined by numerical integration of $\dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u})$ for a given $\boldsymbol{u}$. In general, this can be viewed as an *incremental simulator* that takes an input and produces a trajectory segment according to $\dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u})$.

7.5.3 Decoupled Approach

A popular paradigm for trajectory planning and other problems that involve dynamics is to decouple the problem into first planning a path and then computing a timing function along the path by performing a search in the space spanned by $(s, \dot{s})$, in which s is the path

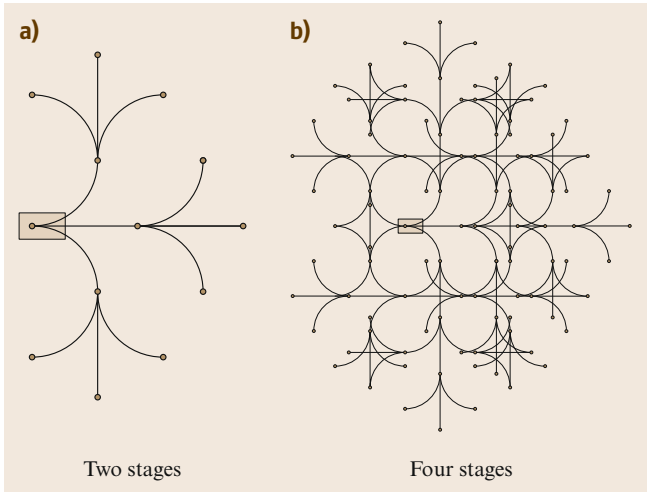


Fig. 7.12 **(a)** A reachability tree for the Dubins car with three actions. **(b)** A 2-stage tree is shown. The k-th stage produces 3^k new vertices

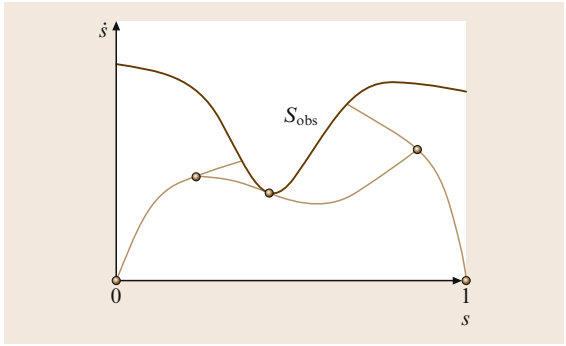


Fig. 7.13 An illustration of the bang–bang approach to computing a time-optimal trajectory. The solution trajectory is obtained by connecting the dots

parameter and $\dot{s}$ is its first derivative. This leads to a diagram such as the one shown in Fig. 7.13, in which the upper region S_{obs} must be avoided because the corresponding motion of the mechanical system violates the differential constraints. Most methods are based on early work in [7.68, 69], and determine a *bang–bang -control*, which means that they switch between accelerating and decelerating at full speed. This applies to determining time-optimal trajectories (optimal once constrained to the path). Dynamic programming can be used for more general problems [7.70].

For some problems and nonholonomic systems, *steering methods* have been developed to solve the two-point boundary value problem efficiently [7.62, 71]. This means that, for any pair of states, a trajectory that ignores obstacles but satisfies the differential constraints can be obtained. Moreover, for some systems,

Avidbots Ex1016
Page 195 of 2154

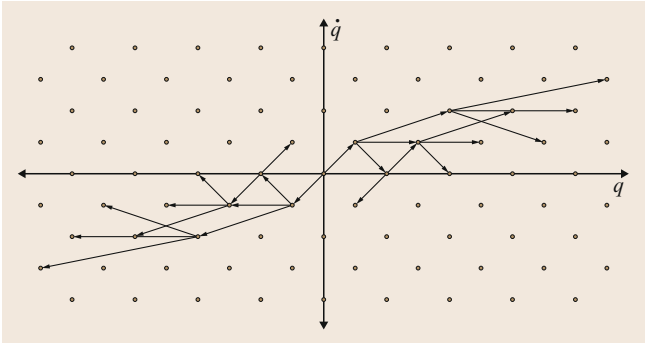


Fig. 7.14 Reachability graph from the origin, shown after three stages (the true edges are actually parabolic arcs when acceleration or deceleration occurs). Note that a lattice is obtained, but the distance traveled in one stage increases as $|\dot{\boldsymbol{q}}|$ increases

the complete set of optimal trajectories has been characterized [7.72, 73]. These control-based approaches enable straightforward adaptation of the sampling-based roadmap approach [7.74, 75]. One decoupled approach is to first plan a path that ignores differential constraints, and then incrementally transform it into one that obeys the constraints [7.62, 76].

7.5.4 Kinodynamic Planning

Due to the great difficulty of planning under differential constraints, many successful planning algorithms that address kinodynamic problems directly in the phase space X are sampling based.

Sampling-based planning algorithms proceed by exploring one or more reachability trees. Many parallels can be drawn with searching on a grid; however, reachability trees are more complicated because they do not necessarily involve a regular lattice structure. The vertex set of reachability trees is dense in most cases. It is therefore not clear how to search a bounded region exhaustively at a fixed resolution. It is also difficult to design approaches that behave like a multiresolution grid, in which refinements can be made arbitrarily to ensure resolution completeness.

Many algorithms attempt to convert the reachability tree into a lattice. This is the basis of the original kinodynamic planning work [7.63], in which the discrete-time approximation to the double integrator, $\ddot{\boldsymbol{q}} = \boldsymbol{u}$, is forced onto a lattice as shown in Fig. 7.14. This enables an approximation algorithm to be developed that solves the kinodynamic planning problem in time polynomial in the approximation quality $1/\epsilon$ and the number of primitives that define the obstacles. Generalizations of the methods to fully actuated systems are described in [7.7]. Surprisingly, it is even possible to obtain a lattice for some underactuated, nonholonomic systems [7.77].

If the reachability tree does not form a lattice, then one approach is to force it to behave as a lattice by imposing a regular cell decomposition over X (or C), and allowing no more than one vertex per cell to be expanded in the reachability graph (Fig. 7.15). This idea was introduced in [7.78]. In their version of this approach, the reachability graph is expanded by dynamic programming. Each cell is initially marked as being in collision or being collision free, but not yet visited. As cells are visited during the search, they become marked as such. If a potential new vertex lands in a visited cell, it is not saved. This has the effect of pruning the reachability tree.

Other related approaches do not try to force the reachability tree onto a lattice. RRTs were designed to expand the tree in a way that is biased toward covering as much new territory as possible in each iteration [7.79]. Planners that are based on the concept of expansive trees attempt to control the density of ver-

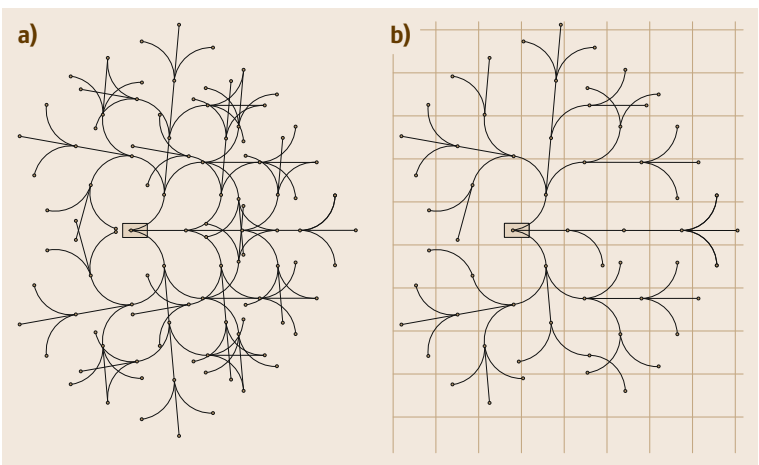


Fig. 7.15 **(a)** The first four stages of a dense reachability graph for the Dubins car. **(b)** One possible search graph, obtained by allowing at most one vertex per cell. Many branches are pruned away. In this simple example there are no cell divisions along the θ-axis

Avidbots Ex1016
Page 196 of 2154

tices in the tree by analyzing neighborhoods [7.44]. The path-directed subdivision tree planner expands a tree, while building an adaptive subdivision of the state space, so as to avoid resampling the same regions of the space [7.47, 80]. Such approaches can be biased to accelerate the expansion of the tree towards a goal, while still providing the weaker probabilistic completeness guarantee [7.48]. VIDEO 24 provides an example of the use of a tree-based planner together with a physics-engine which accounts for the constraints.

7.6 Extensions and Variations

A brief overview of other important extensions to the basic motion planning problem are presented in this section.

7.6.1 Closed Kinematic Chains

In many cases, the robot may be consist of links that form closed loops. This arises in many important applications, for example, if two arms grasp an object then a loop is formed and a humanoid robot forms a loop if both legs touch the ground. For *parallel robots*, loops are intentionally designed into the robot [7.81]; a classic example is the Stewart–Gough platform. To model closed-chain problems, the loops are broken so that a kinematic tree of links is obtained. The main complication is that constraints on C of the form $h(\boldsymbol{q}) = 0$ are introduced, which require that the loops are maintained. This causes great trouble for most planning algorithms because without loops a parameterization of C was available. The closure constraints restrict the planning to a lower-dimensional subset of C for which no parameterization is given. Computing a parameterization is generally difficult or impossible [7.82], although there has been recent progress for some special cases [7.83].

Sampling-based approaches can generally be adapted to handle closed chains. The main difficulty is that the samples $\alpha(i)$ over C are unlikely to be configurations that satisfy closure. In [7.84], both RRTs and PRMs were adapted to closed chains. RRTs performed much better because a costly optimization was required in the PRM to move samples onto the closure subspace; RRTs on the other hand do not require samples to lie in this subspace. By decomposing chains into active and passive links, followed by inverse kinematics computations, performance was dramatically improved for PRMs in [7.85]. This idea was further improved by the introduction of the *random loop generator* (RLG). Based on this, some of the most challenging closed-chain planning problems ever solved appear in [7.86].

7.6.2 Manipulation Planning

In most forms of motion planning, the robot is not allowed to touch obstacles. Suppose instead that it is expected to interact with its environment by manipulating objects. The goal may be to bring an object from one place to another, or to rearrange a collection of objects. This leads to a kind of hybrid motion planning problem, which mixes discrete and continuous spaces. There are discrete modes that correspond to whether the robot is carrying a part [7.87]. In the *transit mode*, the robot moves toward a part. In the *transfer mode*, it carries the part. Transitions between modes require meeting specific grasping and stability requirement. One important variant of manipulation planning is *assembly planning*, in which the goal is to fit a collection of pieces together to make an assembled product [7.88]. Most motion planning work makes limiting assumptions on the kinds of interaction that can occur between the robot and the objects. For richer models of manipulation, see [7.89].

7.6.3 Time-Varying Problems

Suppose that the workspace contains moving obstacles whose trajectories are specified as a function of time. Let $T \subset \mathbb{R}$ denote the *time interval*, which may be *bounded* or *unbounded*. A state X is defined as $X = C \times T$, in which C is the usual C-space of the robot. The obstacle region in X is characterized as

$$X_{\text{obs}} = \{(\boldsymbol{q}, t) \in X \mid \mathcal{A}(\boldsymbol{q}) \cap \mathcal{O}(t) \neq \emptyset\} , \tag{7.5}$$

in which $\mathcal{O}(t)$ is a time-varying obstacle. Many planning algorithms can be adapted to X, which has only one more dimension than C. The main complication is that time must always increase along a path through X.

For the easiest version of the problem, there is no bound on the robot speed. In this case, virtually any sampling-based algorithm can be adapted. Incremental searching and sampling methods apply with little modification, except that paths are directed so that forward time progress is made. Using bidirectional approaches is more difficult for time-varying problems because the goal is usually not a single point due to the time dependency. Sampling-based roadmaps can be adapted; however, a *directed roadmap* is needed, in which every edge must be directed to yield a time-monotonic path.

Avidbots Ex1016
Page 197 of 2154

If the motion model is *algebraic* (i. e., expressed with polynomials) then X_{obs} is semi-algebraic. This enables cylindrical algebraic decomposition to apply. If X_{obs} is polyhedral, as depicted in Fig. 7.16, then vertical decomposition can be used. It is best to first sweep the plane along the T-axis, stopping at the critical times when the linear motion changes.

There has been no consideration so far of the speed at which the robot must move to avoid obstacles. It is obviously impractical in many applications if the solution requires the robot to move arbitrarily fast. One step towards making a realistic model is to enforce a bound on the speed of the robot. Unfortunately, the problem is considerably more difficult. Even for piecewise-linear motions of obstacles in the plane, the problem has been established to be PSPACE-hard [7.90]. A complete algorithm based on the shortest-path roadmap is presented in [7.91].

An alternative to defining the problem in $C \times T$ is to decouple it into a *path planning* part and a *motion timing* part. A collision-free path in the absence of obstacles is first computed. A search in a 2-D space is then performed to determine the *timing function* (or *time scaling*) for the path.

7.6.4 Multiple Robots

A simple extension to the basic motion planning problem can be made to handle multibody robots by including robot self-intersections; however, it is important to specify the pairs of bodies for which collision is unacceptable, for example, consecutive links in a robot arm are allowed to touch.

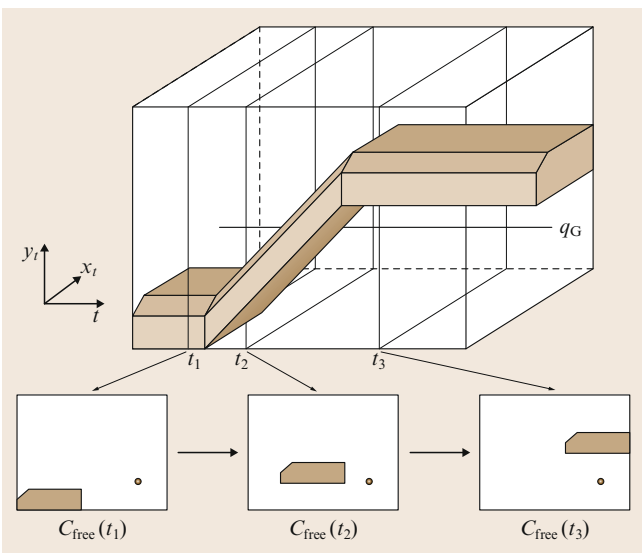


Fig. 7.16 A time-varying example with linear obstacle motion

Substantial attention has been devoted to the problem of planning for multiple robots (VIDEO 21 and VIDEO 22). Suppose there are m robots. A state space is defined that considers the configurations of all robots simultaneously,

$$X = C^1 \times C^2 \times \cdots \times C^m. \tag{7.6}$$

A state $\boldsymbol{x} \in X$ specifies all robot configurations, and may be expressed as $\boldsymbol{x} = (\boldsymbol{q}^1, \boldsymbol{q}^2, \ldots, \boldsymbol{q}^m)$. The dimension of X is N, which is $N = \sum_{i=1}^{m} \dim(C^i)$.

There are two sources of obstacle regions in the state space: (1) *robot–obstacle* collisions, and (2) *robot–robot* collisions. For each i such that $1 \leq i \leq m$, the subset of X that corresponds to robot $\mathcal{A}^i$ in collision with the obstacle region $\mathcal{O}$ is

$$X^i_{obs} = \{\boldsymbol{x} \in X \mid \mathcal{A}^i(\boldsymbol{q}^i) \cap \mathcal{O} \neq \emptyset\}. \tag{7.7}$$

This models the robot–obstacle collisions.

For each pair, $\mathcal{A}^i$ and $\mathcal{A}^j$, of robots, the subset of X that corresponds to $\mathcal{A}^i$ in collision with $\mathcal{A}^j$ is

$$X^{ij}_{obs} = \{\boldsymbol{x} \in X \mid \mathcal{A}^i(\boldsymbol{q}^i) \cap \mathcal{A}^j(\boldsymbol{q}^j) \neq \emptyset\}. \tag{7.8}$$

Both (7.7) and (7.8) will be combined in (7.9) to yield X_{obs}. The obstacle region in X is

$$X_{obs} = \left(\bigcup_{i=1}^{m} X^i_{obs} \right) \bigcup \left(\bigcup_{ij,\ i \neq j} X^{ij}_{obs} \right). \tag{7.9}$$

Once these definitions have been made, any general-purpose planning algorithm can be applied because X and X_{obs} appear no different from C and C_{obs}, except that the dimension N may be very high. Approaches that plan directly in X are called *centralized*. The high dimensionality of X motivates the development of *decoupled* approaches that handle some aspects of the planning independently for each robot. Decoupled approaches are usually more efficient, but this usually comes at the expense of sacrificing completeness. An early decoupled approach is *prioritized planning* [7.92, 93], in which a path and timing function is computed for the i-th robot while treating the first $i-1$ robots as moving obstacles as they follow their paths. Another decoupled approach is *fixed-path* coordination [7.94], in which the paths are planned independently for each robot, and then their timing functions are determined by computing a collision-free path through an m-dimensional *coordination space*. Each axis in this space corresponds to the domain of the path of one robot. Fig. 7.17 shows an example. The idea has been generalized to coordination on roadmaps [7.95, 96].

Avidbots Ex1016
Page 198 of 2154

7.6.5 Uncertainty in Predictability

If the execution of the plan is not predictable, then feedback is needed. The uncertainty may be modeled either *implicitly*, which means that the plan is able to respond to unexpected future configurations, or *explicitly*, which means that the uncertainty is precisely characterized and analyzed in the development of a plan. Potential-function-based approaches are one way of achieving feedback motion planning.

A plan can be represented as a vector field over C_{free}, in which each vector indicates the required velocity. The integral curves of the field should flow into the goal without leaving C_{obs}. If dynamics are a concern, then the vector field can be tracked by an *acceleration-based control model*

$$\boldsymbol{u} = K(f(\boldsymbol{q}) - \dot{\boldsymbol{q}}) + \nabla_{\dot{\boldsymbol{q}}} f(\boldsymbol{q}) \,, \tag{7.10}$$

in which K is a scalar *gain constant*. Alternatively, a vector field may be designed directly on the phase space, X; however, there are not methods to compute such fields efficiently under general conditions. This can also be considered as a feedback control problem with implicit, nonlinear constraints on X.

If the uncertainty is modeled explicitly, then a *game against nature* is obtained, in which the uncertainty is caused by a special decision maker called *nature*. The decisions of nature can either be modeled *nondeterministically*, which means that a set of possible actions is specified, or *probabilistically*, which means that a probability distribution or density is specified over the nature actions. Under nondeterministic uncertainty, *worst-case analysis* is usually performed to select a plan; under probabilistic uncertainty, *expected-case analysis* is usually performed. Numerous approaches exist for such problems, including value iteration, Dijkstra-like algorithms, and reinforcement learning algorithms [7.7].

7.6.6 Sensing Uncertainty

Consider solving tasks such as localization, map building, manipulation, target tracking, and pursuit-evasion (hide-and-seek) with limited sensing. If the current configuration or state is not known during execution, then the problem may appear quite different. Information is obtained from sensors, and the planning problem naturally lives in an *information space* or I-*space* [7.7, Chap. 11]. The state may include the configuration, velocities, or even the map of the environment (e.g., obstacles). The most basic I-space is the set of all histories that can be obtained during execution, based on all sensing observations, actions previously applied,

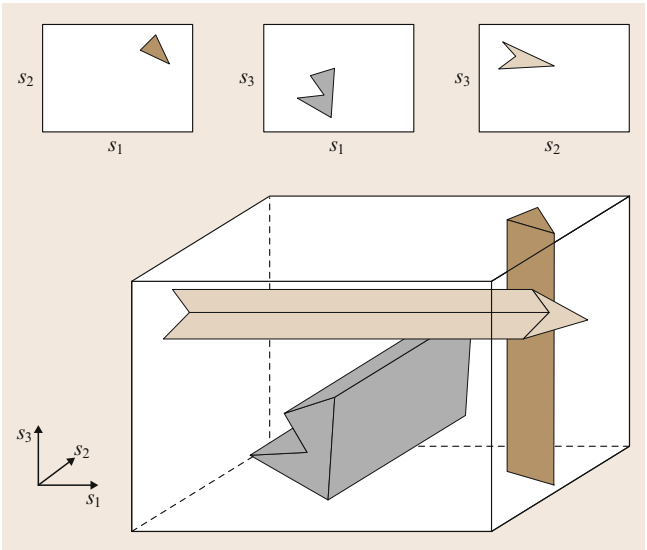


Fig. 7.17 The obstacles that arise from coordinating m robots are always cylindrical. The set of all $\frac{1}{2}m(m-1)$ axis-aligned 2-D projections completely characterizes X_{obs}

and the initial conditions. The goal in developing efficient algorithms in this context is to determine information mappings that reduce the I-space size or complexity so that plans that can be computed that use *information feedback*. The traditional way to use the information state is for estimating the state. This is sufficient for solving many tasks, but it is often not necessary. It may be possible to design and execute successful plans without ever knowing the current state. This can lead to more robust robot systems which may also be cheaper to manufacture due to weaker sensing requirements.

Two important families of I-spaces are *nondeterministic* and *probabilistic*. A nondeterministic information state (I-state) is a set of states that are possible given the available history of sensor observations and actions applied during execution. The nondeterministic I-space is the set of all possibilities. Similarly, a probabilistic I-state is a probability density function over the state space, conditioned on the available history. The probabilistic I-space is often called the *belief space*, which represents the set of all probability density functions. Both filtering and planning over these spaces remains a topic of active research. One of most useful and classical results is the *Kalman filter* [7.97], for which the belief space reduces to Gaussians, allowing it to be completely parametrized by mean and covariance of state. Many approaches to reasoning in these I-spaces attempt to reduce its complexity, through combinatorial reasoning in the case of nondeterministic I-spaces [7.98] and through approximations,

Avidbots Ex1016
Page 199 of 2154

sampling, and dimensionality reduction techniques in belief spaces [7.99–103]. For example, a sampling-based roadmap can be constructed directly in the belief space [7.104].

7.6.7 Optimal Planning

In most formulations of planning, computing an optimal solution is given little or no importance. Several factors have contributed to this trend. One of the most fundamental is that a natural criterion of optimality often does not exist. Unlike control theory, where optimality is a central goal, the main task in planning is to avoid obstacles. Imagine walking quickly through a furniture store. Moving along the shortest possible path would cause you to touch the corners of obstacles, which might not be desirable. You could try to maximize clearance, but this causes longer paths, which again might not be desirable. Another factor is that solutions produced by path planning algorithms tend to not be excessively long, especially after some quick post-processing in the case of sampling-based planners. Finally, the computational complexity of the optimal planning problem is typically worse than its feasible (not necessarily optimal) counterpart. One of the most notable exceptions is planning in field robotics (outdoor vehicles in unstructured terrain), for which cost function determines navigability at each point, leading to the well-known family of *D* algorithms* [7.105, 106].

In spite of these issues, several useful approaches and interesting ideas have emerged. As mentioned in Sect. 7.4.1, the shortest-path roadmap is an effective multiple-query approach in the case of a 2-D, polygonal C-space. Alternatively, the *continuous Dijkstra* method provides an effective single-query approach by propagating wavefronts that correspond to level sets of the optimal cost from the initial configuration [7.107, 108]. The wavefronts are propagated combinatorially, stopping only at critical events, leading to an exact, optimal solution. In the case of 3-D polyhedral C-spaces, the shortest path problem already becomes PSPACE-hard [7.15]. However, algorithms that produce approximately optimal solutions exist [7.109–111], and are useful in C-spaces of several dimensions. Dijkstra-like approaches can be adapted to include various forms of uncertainty and differential constraints, and are all derived in some way from *value iteration* methods introduced by Bellman in the 1950s. See Chaps. 7, 10, and 14 of [7.7] for more discussion. Pushing into even higher dimensions, recent sampling-based planning methods have produced asymptotically optimal versions RRTs [7.112] and PRMs [7.112, 113], which have been shown to produce paths that improve in quality as time progresses.

7.7 Advanced Issues

We cover here a series of more advanced issues, such as topics from topology and sampling theory, and how they influence the performance of motion planners. The last section is devoted to computational algebraic geometry techniques that achieve completeness in the general case. Rather than being a practical alternative, these techniques serve as an upper bound on the best asymptotic running time that could be obtained.

7.7.1 Topology of Configuration Spaces

Manifolds

One reason that the topology of a C-space is important is because it affects its representation. Another reason is that, if a path-planning algorithm can solve problems in a topological space, then that algorithm may carry over to topologically equivalent spaces.

The following definitions are important in order to describe the topology of C-space. A map $\phi : S \to T$ is called a *homeomorphism* if ϕ is a bijection and both ϕ and ϕ^{-1} are continuous. When such a map exists, S and T are said to be homeomorphic. A set S is an n-dimensional *manifold* if it is locally homeomorphic to $\mathbb{R}^n$, meaning that each point in S possesses a neighborhood that is homeomorphic to $\mathbb{R}^n$. For more details, see [7.114, 115].

In the vast majority of motion planning problems, the configuration space is a manifold. An example of a C-space that is not a manifold is the closed unit square: $[0, 1] \times [0, 1] \subset \mathbb{R}^2$, which is a manifold with boundary obtained by pasting the one-dimensional boundary on the two-dimensional open set $(0, 1) \times (0, 1)$. When a C-space is a manifold, then we can represent it with just n parameters, in which n is the dimension of the configuration space. Although an n-dimensional manifold can be represented using as few as n parameters, due to constraints it might be easier to use a representation that has higher number of parameters, e.g., the unit circle $\mathbb{S}^1$ can be represented as $\mathbb{S}^1 = \{(x, y) | x^2 + y^2 = 1\}$ by embedding $\mathbb{S}^1$ in $\mathbb{R}^2$. Similarly, the torus T^2 can be embedded in $\mathbb{R}^3$.

Representation

Embeddings into higher-dimensional spaces can facilitate many C-space operations. For example, the orientation of a rigid body in space can be represented by

a $n \times n$ matrix of real numbers. The n^2 matrix entries must satisfy a number of smooth equality constraints, making the manifold of such matrices a *submanifold* of $\mathbb{R}^{n^2}$. One advantage is that these matrices can be multiplied to get another matrix in the manifold. For example, the orientation of a rigid-body in n-dimensional space ($n = 2$ or 3) is described by the set $SO(n)$, the set of all $n \times n$ rotation matrices. The position and orientation of a rigid body is represented by the set $SE(n)$, the set of all $n \times n$ homogeneous transformation matrices. These matrix groups can be used to (1) represent rigid-body configurations, (2) change the reference frame for the representation of a configuration, and (3) displace a configuration.

There are numerous parameterizations of $SO(3)$ [7.116] but unit quaternions correctly preserve the C-space topology as $\mathbb{S}^1$ represents 2-D rotations. Quaternions were introduced in Chap. 2. There is, however, a two-to-one correspondence between unit quaternions and 3-D rotation matrices. This causes a topological issue that is similar to the equivalence of 0 and 2π for 2-D rotations. One way to account for this is to declare antipodal (opposite) points on $\mathbb{S}^3$ to be equivalent. In planning, only the upper hemisphere of $\mathbb{S}^3$ is needed, and paths that cross the equator instantly reappear on the opposite side of $\mathbb{S}^3$, heading back into the northern hemisphere. In topology, this is called a real projective space: $\mathbb{RP}^3$. Hence, the C-space of a 3-D body capable only of rotation is $\mathbb{RP}^3$. If both translation and rotation are allowed, then $SE(3)$, the set of all 4×4 homogeneous transformation matrices, yields

$$C = \mathbb{R}^3 \times \mathbb{RP}^3 \,, \tag{7.11}$$

which is six dimensional. A configuration $\boldsymbol{q} \in C$ can be expressed using quaternions with seven coordinates, (x, y, z, a, b, c, d), in which $a^2 + b^2 + c^2 + d^2 = 1$. More examples can be found in Table 7.1.

7.7.2 Sampling Theory

Since the most successful paradigm for motion planning today is the sampling-based framework, presented in Sect. 7.3, sampling theory becomes relevant to the motion planning problem.

Table 7.1 Some common robots and their C-spaces

Type of robot	C-space representation
Mobile robot translating in the plane	$\mathbb{R}^2$
Mobile robot translating and rotating in the plane	$SE(2)$ or $\mathbb{R}^2 \times S^1$
Rigid body translating in the three-space	$\mathbb{R}^3$
A spacecraft	$SE(3)$ or $\mathbb{R}^3 \times SO(3)$
An n-joint revolute arm	T^n
A planar mobile robot with an attached n-joint arm	$SE(2) \times T^n$

Metrics in Configuration/State Spaces

Virtually all sampling-based methods require some notion of distance on C. For example, the sampling-based roadmap method selects candidate vertices to connect a new configuration given a distance-defined neighborhood. Similarly, the rapidly exploring dense trees expands the tree from the nearest node of the tree to a newly sampled configuration. Usually, a *metric*, $\rho : C \times C \to \mathbb{R}$, is defined, which satisfies the standard axioms: nonnegativity, reflexivity, symmetry, and the triangle inequality.

Two difficult issues that arise in constructing a metric are: (1) the topology of C must be respected, and (2) several different quantities, such as linear and angular displacements, must be compared in some way. To illustrate the second issue, consider defining a metric ρ_z for a space constructed as $Z = X \times Y$ as

$$\begin{aligned} \rho_z(z, z') &= \rho_z(x, y, x', y') \\ &= c_1 \rho_x(x, x') + c_2 \rho_y(y, y') \,. \end{aligned} \tag{7.12}$$

Above, c_1 and c_2 are arbitrary positive constants that indicate the relative weights of the two components. For a 2-D rotation, θ_i, expressed as $a_i = \cos\theta_i$ and $b_i = \sin\theta_i$, a useful metric is

$$\rho(a_1, b_1, a_2, b_2) = \cos^{-1}(a_1 a_2 + b_1 b_2) \,. \tag{7.13}$$

The 3-D equivalent is obtained by defining

$$\rho_0(\boldsymbol{h}_1, \boldsymbol{h}_2) = \cos^{-1}(a_1 a_2 + b_1 b_2 + c_1 c_2 + d_1 d_2) \,, \tag{7.14}$$

in which each $\boldsymbol{h}_i = (a_i, b_i, c_i, d_i)$ is a unit quaternion. The metric is defined as $\rho(\boldsymbol{h}_1, \boldsymbol{h}_2) = \min(\rho_0(\boldsymbol{h}_1, \boldsymbol{h}_2), \rho_0(\boldsymbol{h}_1, -\boldsymbol{h}_2))$, by respecting the required identification of antipodal points. This computes the shortest distance in $\mathbb{R}^4$, for a path constrained to the unit sphere.

In some algorithms, defining volume on C may also be important. In general, this leads to a *measure space*, for which the volume function (called the *measure*) must satisfy axioms that resemble the probability axioms, but without normalization. For transformation groups, one must be careful to define volumes in a way that is invariant with respect to transformations. Such volumes are called *Haar measures*. Defining volumes via balls using the metric definitions (7.13) and (7.14) actually satisfy this concern.

Probabilistic Versus Deterministic Sampling

The C-space may be sampled *probabilistically* or *deterministically*. Either way, the requirement is usually that a dense sequence α of samples is obtained.

Avidbots Ex1016
Page 201 of 2154

This means that, in the limit as the number of samples tends to infinity, the samples become arbitrarily close to every point in C. For probabilistic sampling, this denseness (with probability one) ensures *probabilistic completeness* of a planning algorithm. For deterministic sampling, it ensures *resolution completeness*, which means that, if a solution exists, the algorithm is guaranteed to find it; otherwise, it may run forever.

For probabilistic sampling, samples are selected randomly over C, using a uniform probability density function. To obtain uniformity in a meaningful way, the Haar measure should be used. This is straightforward in many cases; $SO(3)$ however is tricky. A uniform (with respect to Haar measure) random quaternion is selected as follows. Choose three points $u_1, u_2, u_3 \in [0, 1]$ uniformly at random, and let [7.117]

$$\boldsymbol{h} = \left(\sqrt{1-u_1} \sin 2\pi u_2,\ \sqrt{1-u_1} \cos 2\pi u_2\ , \right.$$
$$\left. \sqrt{u_1} \sin 2\pi u_3,\ \sqrt{u_1} \cos 2\pi u_3 \right)\ . \tag{7.15}$$

Even though random samples are uniform in some sense, they are also required to have some irregularity to satisfy statistical tests. This has motivated the development of deterministic sampling schemes that offer better performance [7.118]. Instead of being concerned with randomness, deterministic sampling techniques are designed to optimize criteria, such as *discrepancy* and *dispersion*. Discrepancy penalizes regularity in the sample, which frequently causes trouble in numerical integration. Dispersion gives the radius of the largest empty (not containing samples) ball. Thus, driving dispersion down quickly means that the whole space is explored quickly. Deterministic samples may be *irregular* neighborhood structure (appearing much like random samples), or *regular* neighborhood structure, which means that points are arranged along a grid or lattice. For more details in the context of motion planning, see [7.7].

7.7.3 Computational Algebraic Geometry Techniques

Sampling-based algorithms provide good practical performance at the expense of achieving only a weaker form of completeness. On the other hand, complete algorithms, which are the focus of this section, are able to deduce that there is no solution to a planning problem.

Complete algorithms are able to solve virtually any motion planning problem as long as C_{obs} is represented by patches of algebraic surfaces. Formally, the model must be *semi-algebraic*, which means that it is formed from unions and intersections of roots of multivariate polynomials in q, and for computability, the polynomials must have rational coefficients (otherwise roots may not have finite representations). The set of all roots to polynomials with rational coefficients is called *real algebraic numbers* and has many nice computational properties. See [7.12, 119–121] for more information on the exact representation and calculation with real algebraic numbers. For a gentle introduction to algebraic geometry, see [7.82].

To use techniques based on algebraic geometry, the first step is to convert the models into the required polynomials. Suppose that the models, the robot, $\mathcal{A}$, and the obstacles $\mathcal{O}$ are semi-algebraic (this includes polyhedral models). For any number of attached 2-D or 3-D bodies, the kinematic transformations can be expressed using polynomials. Since polynomial transformations of polynomials yield polynomials, the transformed robot model is polynomial. The algebraic surfaces that comprise C_{obs} are computed by carefully considering all contact types, which characterize all ways to pair a robot feature (faces, edges, vertices) with an obstacle feature [7.6, 7, 9, 122]. This step already produces too many model primitives to be useful in most applications.

Once the semi-algebraic representation has been obtained, powerful techniques from algebraic geometry can be exploited. One of the most widely known algorithms, *cylindrical algebraic decomposition* [7.119, 123, 124], provides the information needed to solve the motion planning problem. It was originally designed to determine whether *Tarski sentences*, which involve quantifiers and polynomials, are satisfiable, and to find an equivalent expression that does not involve quantifiers. The decomposition produces a finite set of cells over which the signs of the polynomials remain fixed. This enables a systematic approach to satisfiability and quantifier elimination. It was recognized by *Schwartz* and *Sharir* [7.121] that it also solves motion planning.

The method is conceptually simple, but there are many difficult technical details. The decomposition is called cylindrical because the cells are organized into vertical columns of cells, see Fig. 7.18 for a 2-D example. There are two kinds of critical events, shown in Fig. 7.19. At critical points, rays are extended indefinitely in both vertical directions. The decomposition differs from the vertical decomposition in Fig. 7.7 because there the rays were only extended until the next obstacle was hit. Here, columns of cells are obtained.

In n dimensions, each column represents a chain of cells. The first and last cells are n-dimensional and unbounded. The remaining cells are bounded and alternate between being $(n-1)$-dimensional and n-dimensional.

Avidbots Ex1016
Page 202 of 2154

The bounded n-dimensional cells are bounded above and below by the roots of single multivariate polynomials. This makes it simple to describe the cells and their connectivity. To compute this cell decomposition, the algorithm constructs a cascading chain of projections. In the first step, C_{obs} is projected from $\mathbb{R}^n$ to $\mathbb{R}^{n-1}$. This is followed by a projection into $\mathbb{R}^{n-2}$. This repeats until $\mathbb{R}$ is obtained with a univariate polynomial that encodes the places at which all critical boundaries need to be placed. In a second phase of the algorithm, a series of liftings is performed. Each lifting takes the polynomials and cell decomposition over $\mathbb{R}^i$ and lifts them via columns of cells to $\mathbb{R}^{i+1}$. A single lifting is illustrated in Fig. 7.18b. The running time of the full algorithm depends on the particular methods used to perform the algebraic computations. The total running time required to use cylindrical algebraic decomposition for motion planning is bounded by $(md)^{O(1)^n}$, in which m is the number of polynomials to describe C_{obs} (a huge number), and d is the maximum algebraic degree. (It may seem odd for $O(\cdot)$ to appear in the middle of an expression. In this context, it means that there exists some $c \in [0, \infty)$ such that the running time is bounded by $(md)^{c^n}$. Note that another O is not necessary in the front of the whole formula.) The main point to remember is that the algorithm is doubly exponential in the dimension of C (even the number of cells is doubly exponential).

Although performing the cylindrical decomposition is sufficient for solving motion planning, it computes more information than is necessary. This motivates *Canny*'s roadmap algorithm [7.12], which produces a roadmap directly from the semi-algebraic set, rather than constructing a cell decomposition along the way. Since there are doubly exponentially many cells in the cylindrical algebraic decomposition, avoiding this construction pays off. The resulting roadmap method of *Canny* solves the motion planning problem in time that is again polynomial in the number of polynomials and polynomial in the algebraic degree, but is only singly exponential in dimension [7.12].

The basic idea is to find silhouette curves in $\mathbb{R}^2$ of C_{obs} in $\mathbb{R}^n$. The method finds zero-dimensional critical points and one-dimensional critical curves. The critical curves become roadmap edges, and the critical points are places at which the algorithm recursively finds silhouettes of $(n-1)$-dimensional slices of C_{obs}. These contribute more critical points and curves. The curves are added to the roadmap, and the algorithm recurses again on the critical points. The recursive iterations terminate at $n = 2$. Canny showed that the resulting union of critical curves preserves the connectivity of C_{obs} (and hence, C_{free}). Some of the technical issues are: the algorithm works with a stratification of C_{obs} into manifolds; there are strong general position assumptions that are hard to meet; paths are actually considered along the boundary of C_{free}; and the method does not produce a parameterized solution path. For improvements to Canny's algorithm and many other important details, see [7.119].

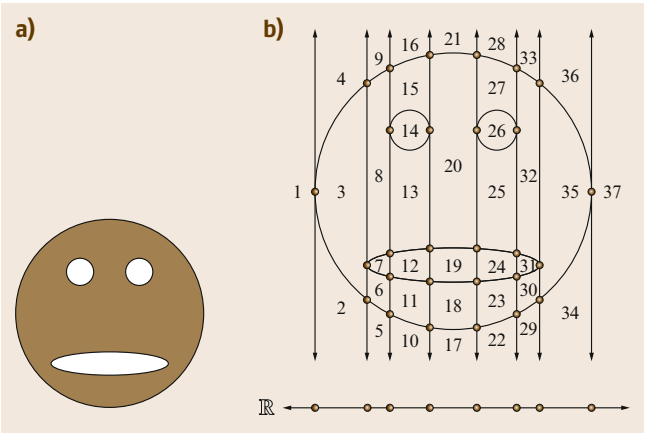


Fig. 7.18 **(a)** A face modeled with four algebraic primitives, and **(b)** a cylindrical algebraic decomposition of the face

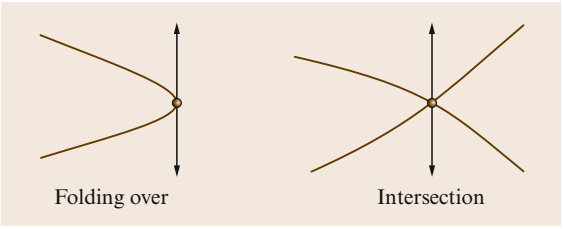


Fig. 7.19 Critical points occur either when the surface folds over in the vertical direction or when surfaces intersect

7.8 Conclusions and Further Reading

The brief survey given here hardly does justice to motion planning, which is a rich and active research field. For more details, we recommend consulting two recent textbooks [7.5, 7]. In addition, see the classic textbook of *Latombe* [7.6], the classic papers in [7.4], and the recent surveys in [7.2, 3]. Furthermore, consult the related handbook chapters that were indicated throughout this chapter.

Avidbots Ex1016
Page 203 of 2154

Video-References

VIDEO 17 Powder transfer task using demonstration-guided motion planning
available from http://handbookofrobotics.org/view-chapter/07/videodetails/17

VIDEO 21 Simulation of a large crowd
available from http://handbookofrobotics.org/view-chapter/07/videodetails/21

VIDEO 22 Motion planning in multi-robot scenario
available from http://handbookofrobotics.org/view-chapter/07/videodetails/22

VIDEO 23 Alpha puzzle
available from http://handbookofrobotics.org/view-chapter/07/videodetails/23

VIDEO 24 Kinodynamic motion planning for a car-like robot
available from http://handbookofrobotics.org/view-chapter/07/videodetails/24

References

7.1 J.H. Reif: Complexity of the mover's problem and generalizations, IEEE Symp. Found. Comput. Sci. (1979) pp. 421–427

7.2 H.H. Gonzalez-Banos, D. Hsu, J.C. Latombe: Motion planning: Recent developments. In: *Automous Mobile Robots: Sensing, Control, Decision-Making and Applications*, ed. by S.S. Ge, F.L. Lewis (CRC, Boca Raton 2006)

7.3 S.R. Lindemann, S.M. LaValle: Current issues in sampling-based motion planning, 11th Int. Symp. Robotics Res. (Springer, Berlin, Heidelberg 2005) pp. 36–54

7.4 J.T. Schwartz, M. Sharir: A survey of motion planning and related geometric algorithms, Artif. Intell. J. **37**, 157–169 (1988)

7.5 H. Choset, K.M. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L.E. Kavraki, S. Thrun: *Principles of Robot Motion: Theory, Algorithms, and Implementations* (MIT, Cambridge 2005)

7.6 J.C. Latombe: *Robot Motion Planning* (Kluwer, Boston 1991)

7.7 S.M. LaValle: *Planning Algorithms* (Cambridge Univ. Press, Cambridge 2006)

7.8 S. Udupa: Collision detection and avoidance in computer controlled manipulators, Ph.D. Thesis (Dept. of Electical Engineering, California Institute of Technology, Pasadena 1977)

7.9 T. Lozano-Pérez: Spatial planning: A configuration space approach, IEEE Trans. Comput. C **32**(2), 108–120 (1983)

7.10 J.T. Schwartz, M. Sharir: On the piano movers' problem: III. Coordinating the motion of several independent bodies, Int. J. Robotics Res. **2**(3), 97–140 (1983)

7.11 J.T. Schwartz, M. Sharir: On the piano movers' problem: V. The case of a rod moving in three-dimensional space amidst polyhedral obstacles, Commun. Pure Appl. Math. **37**, 815–848 (1984)

7.12 J.F. Canny: *The Complexity of Robot Motion Planning* (MIT, Cambridge 1988)

7.13 D. Halperin, M. Sharir: A near-quadratic algorithm for planning the motion of a polygon in a polygonal environment, Discret. Comput. Geom. **16**, 121–134 (1996)

7.14 J.E. Hopcroft, J.T. Schwartz, M. Sharir: On the complexity of motion planning for multiple independent objects: PSPACE-hardness of the warehouseman's problem, Int. J. Robotics Res. **3**(4), 76–88 (1984)

7.15 J. Canny, J. Reif: New lower bound techniques for robot motion planning problems, IEEE Symp. Found. Comput. Sci. (1987) pp. 49–60

7.16 M.C. Lin, J.F. Canny: Efficient algorithms for incremental distance computation, IEEE Int. Conf. Robotics Autom. (1991) pp. 1008–1014

7.17 P. Jiménez, F. Thomas, C. Torras: Collision detection algorithms for motion planning. In: *Robot Motion Planning and Control*, ed. by J.P. Laumond (Springer, Berlin, Heidelberg 1998) pp. 1–53

7.18 M.C. Lin, D. Manocha: Collision and proximity queries. In: *Handbook of Discrete and Computational Geometry*, 2nd edn., ed. by J.E. Goodman, J. O'Rourke (Chapman Hall/CRC, Boca Raton 2004) pp. 787–807

7.19 L.E. Kavraki, P. Svestka, J.C. Latombe, M.H. Overmars: Probabilistic roadmaps for path planning in high-dimensional configuration spaces, IEEE Trans. Robotics Autom. **12**(4), 566–580 (1996)

7.20 N.M. Amato, O.B. Bayazit, L.K. Dale, C. Jones, D. Vallejo: OBPRM: An obstacle-based PRM for 3-D workspaces, Workshop Algorith. Found. Robotics (1998) pp. 155–168

7.21 V. Boor, M.H. Overmars, A.F. van der Stappen: The Gaussian sampling strategy for probabilistic roadmap planners, IEEE Int. Conf. Robotics Autom. (1999) pp. 1018–1023

7.22 C. Holleman, L.E. Kavraki: A framework for using the workspace medial axis in PRM planners, IEEE Int. Conf. Robotics Autom. (2000) pp. 1408–1413

7.23 J.M. Lien, S.L. Thomas, N.M. Amato: A general framework for sampling on the medial axis of the free space, IEEE Int. Conf. Robotics Autom. (2003)

7.24 S.M. LaValle, M.S. Branicky, S.R. Lindemann: On the relationship between classical grid search and probabilistic roadmaps, Int. J. Robotics Res. **23**(7/8), 673–692 (2004)

7.25 T. Siméon, J.-P. Laumond, C. Nissoux: Visibility based probabilistic roadmaps for motion planning, Adv. Robotics **14**(6), 477–493 (2000)

7.26 J. Barraquand, L. Kavraki, J.-C. Latombe, T.-Y. Li, R. Motwani, P. Raghavan: A random sampling

Avidbots Ex1016
Page 204 of 2154

scheme for robot path planning, Proc. Int. Symp. Robotics Res. (1996) pp. 249–264
7.27 A. Ladd, L.E. Kavraki: Measure theoretic analysis of probabilistic path planning, IEEE Trans. Robotics Autom. **20**(2), 229–242 (2004)
7.28 R. Geraerts, M. Overmars: Sampling techniques for probabilistic roadmap planners, Int. Conf. Intell. Auton. Syst. (2004) pp. 600–609
7.29 D. Hsu, T. Jiang, J. Reif, Z. Sun: The bridge test for sampling narrow passages with probabilistic roadmap planners, IEEE Int. Conf. Robotics Autom. (2003) pp. 4420–4426
7.30 R. Bohlin, L. Kavraki: Path planning using lazy PRM, IEEE Int. Conf. Robotics Autom. (2000) pp. 521–528
7.31 B. Burns, O. Brock: Sampling-based motion planning using predictive models, IEEE Int. Conf. Robotics Autom. (2005) pp. 3120–3125
7.32 P. Isto: Constructing probabilistic roadmaps with powerful local planning and path optimization, IEEE/RSJ Int. Conf. Intell. Robots Syst. (2002) pp. 2323–2328
7.33 P. Leven, S.A. Hutchinson: Using manipulability to bias sampling during the construction of probabilistic roadmaps, IEEE Trans. Robotics Autom. **19**(6), 1020–1026 (2003)
7.34 D. Nieuwenhuisen, M.H. Overmars: Useful cycles in probabilistic roadmap graphs, IEEE Int. Conf. Robotics Autom. (2004) pp. 446–452
7.35 S.M. LaValle, J.J. Kuffner: Rapidly-exploring random trees: progress and prospects. In: *Algorithmic and Computational Robotics: New Direction*, ed. by B.R. Donald, K.M. Lynch, D. Rus (A.K. Peters, Wellesley 2001) pp. 293–308
7.36 K.E. Bekris, B.Y. Chen, A. Ladd, E. Plaku, L.E. Kavraki: Multiple query probabilistic roadmap planning using single query primitives, IEEE/RSJ Int. Conf. Intell. Robots Syst. (2003) pp. 656–661
7.37 M. Strandberg: Augmenting RRT-planners with local trees, IEEE Int. Conf. Robotics Autom. (2004) pp. 3258–3262
7.38 J.J. Kuffner, S.M. LaValle: *An Efficient Approach to Path Planning Using Balanced Bidirectional RRT Search*, Techn. Rep. CMU-RI-TR-05-34 Robotics Institute (Carnegie Mellon University, Pittsburgh 2005)
7.39 J. Bruce, M. Veloso: Real-time randomized path planning for robot navigation, IEEE/RSJ Int. Conf. Intell. Robots Syst. (2002) pp. 2383–2388
7.40 E. Frazzoli, M.A. Dahleh, E. Feron: Real-time motion planning for agile autonomous vehicles, AIAA J. Guid. Contr. **25**(1), 116–129 (2002)
7.41 M. Kallmann, M. Mataric: Motion planning using dynamic roadmaps, IEEE Int. Conf. Robotics Autom. (2004) pp. 4399–4404
7.42 A. Yershova, L. Jaillet, T. Simeon, S.M. LaValle: Dynamic-domain RRTs: Efficient exploration by controlling the sampling domain, IEEE Int. Conf. Robotics Autom. (2005) pp. 3867–3872
7.43 D. Hsu, J.C. Latombe, R. Motwani: Path planning in expansive configuration spaces, Int. J. Comput. Geom. Appl. **4**, 495–512 (1999)
7.44 D. Hsu, R. Kindel, J.C. Latombe, S. Rock: Randomized kinodynamic motion planning with moving obstacles. In: *Algorithmic and Computational Robotics: New Directions*, ed. by B.R. Donald, K.M. Lynch, D. Rus (A.K. Peters, Wellesley 2001) pp. 247–264
7.45 G. Sánchez, J.-C. Latombe: A single-query bi-directional probabilistic roadmap planner with lazy collision checking, ISRR Int. Symp. Robotics Res. (2007) pp. 403–413
7.46 S. Carpin, G. Pillonetto: Robot motion planning using adaptive random walks, IEEE Int. Conf. Robotics Autom. (2003) pp. 3809–3814
7.47 A. Ladd, L.E. Kavraki: Fast exploration for robots with dynamics, Workshop Algorithm. Found. Robotics, Amsterdam (2004)
7.48 K.E. Bekris, L.E. Kavraki: Greedy but safe replanning under differential constraints, IEEE Int. Conf. Robotics Autom. (2007) pp. 704–710
7.49 C. O'Dunlaing, C.K. Yap: A retraction method for planning the motion of a disc, J. Algorithms **6**, 104–111 (1982)
7.50 D. Leven, M. Sharir: Planning a purely translational motion for a convex object in two-dimensional space using generalized Voronoi diagrams, Discret. Comput. Geom. **2**, 9–31 (1987)
7.51 M. Sharir: Algorithmic motion planning. In: *Handbook of Discrete and Computational Geometry*, 2nd edn., ed. by J.E. Goodman, J. O'Rourke (Chapman Hall/CRC, Boca Raton 2004) pp. 1037–1064
7.52 N.J. Nilsson: A mobile automaton: An application of artificial intelligence techniques, 1st Int. Conf. Artif. Intell. (1969) pp. 509–520
7.53 J. O'Rourke: Visibility. In: *Handbook of Discrete and Computational Geometry*, 2nd edn., ed. by J.E. Goodman, J. O'Rourke (Chapman Hall/CRC, Boca Raton 2004) pp. 643–663
7.54 B. Chazelle: Approximation and decomposition of shapes. In: *Algorithmic and Geometric Aspects of Robotics*, ed. by J.T. Schwartz, C.K. Yap (Lawrence Erlbaum, Hillsdale 1987) pp. 145–185
7.55 M. de Berg, M. van Kreveld, M. Overmars, O. Schwarzkopf: *Computational Geometry: Algorithms and Applications*, 2nd edn. (Springer, Berlin, Heidelberg 2000)
7.56 J.M. Keil: Polygon decomposition. In: *Handbook on Computational Geometry*, ed. by J.R. Sack, J. Urrutia (Elsevier, New York 2000)
7.57 J.T. Schwartz, M. Sharir: On the piano movers' problem: I. The case of a two-dimensional rigid polygonal body moving amidst polygonal barriers, Commun. Pure Appl. Math. **36**, 345–398 (1983)
7.58 O. Khatib: Real-time obstacle avoidance for manipulators and mobile robots, Int. J. Robotics Res. **5**(1), 90–98 (1986)
7.59 J. Barraquand, J.-C. Latombe: Robot motion planning: A distributed representation approach, Int. J. Robotics Res. **10**(6), 628–649 (1991)
7.60 E. Rimon, D.E. Koditschek: Exact robot navigation using artificial potential fields, IEEE Trans. Robotics Autom. **8**(5), 501–518 (1992)

Avidbots Ex1016
Page 205 of 2154

7.61 J.P. Laumond: Trajectories for mobile robots with kinematic and environment constraints, Int. Conf. Intell. Auton. Syst. (1986) pp. 346–354

7.62 J.P. Laumond, S. Sekhavat, F. Lamiraux: Guidelines in nonholonomic motion planning for mobile robots. In: *Robot Motion Planning and Control*, ed. by J.P. Laumond (Springer, Berlin, Heidelberg 1998) pp. 1–53

7.63 B.R. Donald, P.G. Xavier, J. Canny, J. Reif: Kinodynamic planning, Journal ACM **40**, 1048–1066 (1993)

7.64 C. O'Dunlaing: Motion planning with inertial constraints, Algorithmica **2**(4), 431–475 (1987)

7.65 J. Canny, A. Rege, J. Reif: An exact algorithm for kinodynamic planning in the plane, Discret. Comput. Geom. **6**, 461–484 (1991)

7.66 J. Go, T. Vu, J.J. Kuffner: Autonomous behaviors for interactive vehicle animations, SIGGRAPH/Eurographics Symp. Comput. Animat., Aire-la-Ville (2004) pp. 9–18

7.67 M. Pivtoraiko, A. Kelly: Generating near minimal spanning control sets for constrained motion planning in discrete state spaces, IEEE/RSJ Int. Conf. Intell. Robots Syst. (2005) pp. 3231–3237

7.68 J. Hollerbach: *Dynamic scaling of manipulator trajectories*, Tech. Rep. 700 (MIT, Cambridge 1983)

7.69 K.G. Shin, N.D. McKay: Minimum-time control of robot manipulators with geometric path constraints, IEEE Trans. Autom. Control **30**(6), 531–541 (1985)

7.70 K.G. Shin, N.D. McKay: A dynamic programming approach to trajectory planning of robotic manipulators, IEEE Trans. Autom. Control **31**(6), 491–500 (1986)

7.71 S. Sastry: *Nonlinear Systems: Analysis, Stability, and Control* (Springer, Berlin, Heidelberg 1999)

7.72 D.J. Balkcom, M.T. Mason: Time optimal trajectories for bounded velocity differential drive vehicles, Int. J. Robotics Res. **21**(3), 199–217 (2002)

7.73 P. Souères, J.-D. Boissonnat: Optimal trajectories for nonholonomic mobile robots. In: *Robot Motion Planning and Control*, ed. by J.P. Laumond (Springer, Berlin, Heidelberg 1998) pp. 93–169

7.74 P. Svestka, M.H. Overmars: Coordinated motion planning for multiple car-like robots using probabilistic roadmaps, IEEE Int. Conf. Robotics Autom. (1995) pp. 1631–1636

7.75 S. Sekhavat, P. Svestka, J.-P. Laumond, M.H. Overmars: Multilevel path planning for nonholonomic robots using semiholonomic subsystems, Int. J. Robotics Res. **17**, 840–857 (1998)

7.76 P. Ferbach: A method of progressive constraints for nonholonomic motion planning, IEEE Int. Conf. Robotics Autom. (1996) pp. 2949–2955

7.77 S. Pancanti, L. Pallottino, D. Salvadorini, A. Bicchi: Motion planning through symbols and lattices, IEEE Int. Conf. Robotics Autom. (2004) pp. 3914–3919

7.78 J. Barraquand, J.-C. Latombe: Nonholonomic multibody mobile robots: Controllability and motion planning in the presence of obstacles, Algorithmica **10**, 121–155 (1993)

7.79 S.M. LaValle, J.J. Kuffner: Randomized kinodynamic planning, IEEE Int. Conf. Robotics Autom. (1999) pp. 473–479

7.80 A.M. Ladd, L.E. Kavraki: Motion planning in the presence of drift underactuation and discrete system changes, Robotics Sci. Syst. I, Cambridge (2005) pp. 233–241

7.81 J.-P. Merlet: *Parallel Robots* (Kluwer, Boston 2000)

7.82 D. Cox, J. Little, D. O'Shea: *Ideals, Varieties, and Algorithms* (Springer, Berlin, Heidelberg 1992)

7.83 R.J. Milgram, J.C. Trinkle: The geometry of configuration spaces for closed chains in two and three dimensions, Homol. Homot. Appl. **6**(1), 237–267 (2004)

7.84 J. Yakey, S.M. LaValle, L.E. Kavraki: Randomized path planning for linkages with closed kinematic chains, IEEE Trans. Robotics Autom. **17**(6), 951–958 (2001)

7.85 L. Han, N.M. Amato: A kinematics-based probabilistic roadmap method for closed chain systems. In: *Algorithmic and Computational Robotics: New Directions*, ed. by B.R. Donald, K.M. Lynch, D. Rus (A.K. Peters, Wellesley 2001) pp. 233–246

7.86 J. Cortés: Motion Planning Algorithms for General Closed-Chain Mechanisms, Ph.D. Thesis (Institut National Polytechnique do Toulouse, Toulouse 2003)

7.87 R. Alami, J.-P. Laumond, T. Siméon: Two manipulation planning algorithms. In: *Algorithms for Robotic Motion and Manipulation*, ed. by J.P. Laumond, M. Overmars (A.K. Peters, Wellesley 1997)

7.88 L.E. Kavraki, M. Kolountzakis: Partitioning a planar assembly into two connected parts is NP-complete, Inform. Process. Lett. **55**(3), 159–165 (1995)

7.89 M.T. Mason: *Mechanics of Robotic Manipulation* (MIT, Cambridge 2001)

7.90 K. Sutner, W. Maass: Motion planning among time dependent obstacles, Acta Inform. **26**, 93–122 (1988)

7.91 J.H. Reif, M. Sharir: Motion planning in the presence of moving obstacles, Journal ACM **41**, 764–790 (1994)

7.92 M.A. Erdmann, T. Lozano-Pérez: On multiple moving objects, Algorithmica **2**, 477–521 (1987)

7.93 J. van den Berg, M. Overmars: Prioritized motion planning for multiple robots, IEEE/RSJ Int. Conf. Intell. Robots Syst. (2005) pp. 2217–2222

7.94 T. Siméon, S. Leroy, J.-P. Laumond: Path coordination for multiple mobile robots: A resolution complete algorithm, IEEE Trans. Robotics Autom. **18**(1), 42–49 (2002)

7.95 R. Ghrist, J.M. O'Kane, S.M. LaValle: Pareto optimal coordination on roadmaps, Workshop Algorithm. Found. Robotics (2004) pp. 185–200

7.96 S.M. LaValle, S.A. Hutchinson: Optimal motion planning for multiple robots having independent goals, IEEE Trans. Robotics Autom. **14**(6), 912–925 (1998)

Avidbots Ex1016
Page 206 of 2154

7.97 P.R. Kumar, P. Varaiya: *Stochastic Systems* (Prentice Hall, Englewood Cliffs 1986)

7.98 S.M. LaValle: *Sensing and Filtering: A Fresh Perspective Based on Preimages and Information Spaces*, Foundations and Trends in Robotics (Now Publ., Delft 2012)

7.99 R. Alterovitz, N. Simeon, K. Goldberg: The stochastic motion roadmap: A sampling framework for planning with Markov motion uncertainty, Robotics Sci. Syst. **3**, 233–241 (2007)

7.100 H. Kurniawati, D. Hsu, W.S. Lee: SARSOP: Efficient point-based POMDP planning by approximating optimally reachable belief spaces, Robotics Sci. Syst. (2008)

7.101 J. Pineau, G. Gordon, S. Thrun: Point-based value iteration, Int. Joint Conf. Artif. Intell. (2003) pp. 1025–1032

7.102 R. Platt, R. Tedrake, T. Lozano-Perez, L.P. Kaelbling: Belief space planning assuming maximum likelihood observations, Robotics Sci. Syst. (2010)

7.103 N. Roy, G. Gordon: Exponential family PCA for belief compression in POMDPs, Adv. Neural Inform. Process. Syst. (2003)

7.104 R. He, S. Prentice, N. Roy: Planning in information space for a quadrotor helicopter in a GPS-denied environment, IEEE Int. Conf. Robotics Autom. (2008)

7.105 S. Koenig, M. Likhachev: *D** lite, AAAI Nat. Conf. Artif. Intell. (2002) pp. 476–483

7.106 A. Stentz: Optimal and efficient path planning for partially-known environments, IEEE Int. Conf. Robotics Autom. (1994) pp. 3310–3317

7.107 J. Hershberger, S. Suri: Efficient Computation of Euclidean shortest paths in the plane, IEEE Symp. Found. Comp. Sci. (1995) pp. 508–517

7.108 J.S.B. Mitchell: Shortest paths among obstacles in the plane, Int. J. Comput. Geom. Applic. **6**(3), 309 (1996)

7.109 J. Choi, J. Sellen, C.K. Yap: Precision-sensitive Euclidean shortest path in 3-space, ACM Symp. Comput. Geo. (1995) pp. 350–359

7.110 C.H. Papadimitriou: An algorithm for shortest-path planning in three dimensions, Inform. Process. Lett. **20**(5), 259 (1985)

7.111 D. Yershov, S.M. LaValle: Simplicial Dijkstra and A* algorithms for optimal feedback planning, IEEE/RSJ Int. Conf. Intell. Robots Syst. (2011)

7.112 S. Karaman, E. Frazzoli: Sampling-based algorithms for optimal motion planning, Int. J. Robotics Res. **30**(7), 846 (2011)

7.113 J.D. Marble, K.E. Bekris: Towards small asymptotically near-optimal roadmaps, IEEE Int. Conf. Robotics Autom. (2012)

7.114 W.M. Boothby: *An Introduction to Differentiable Manifolds and Riemannian Geometry*, 2nd edn. (Academic, New York 2003)

7.115 A. Hatcher: *Algebraic Topology* (Cambridge Univ. Press, Cambridge 2002)

7.116 G.S. Chirikjian, A.B. Kyatkin: *Engineering Applications of Noncommutative Harmonic Analysis* (CRC, Boca Raton 2001)

7.117 J. Arvo: Fast random rotation matrices. In: *Graphics Gems III*, ed. by D. Kirk (Academic, New York 1992) pp. 117–120

7.118 H. Niederreiter: *Random Number Generation and Quasi-Monte-Carlo Methods* (Society for Industrial and Applied Mathematics, Philadelphia 1992)

7.119 S. Basu, R. Pollack, M.-F. Roy: *Algorithms in Real Algebraic Geometry* (Springer, Berlin, Heidelberg 2003)

7.120 B. Mishra: Computational real algebraic geometry. In: *Handbook of Discrete and Computational Geometry*, ed. by J.E. Goodman, J. O'Rourke (CRC, Boca Raton 1997) pp. 537–556

7.121 J.T. Schwartz, M. Sharir: On the piano movers' problem: II. General techniques for computing topological properties of algebraic manifolds, Commun. Pure Appl. Math. **36**, 345–398 (1983)

7.122 B.R. Donald: A search algorithm for motion planning with six degrees of freedom, Artif. Intell. J. **31**, 295–353 (1987)

7.123 D.S. Arnon: Geometric reasoning with logic and algebra, Artif. Intell. J. **37**(1-3), 37–60 (1988)

7.124 G.E. Collins: Quantifier elimination by cylindrical algebraic decomposition–twenty years of progress. In: *Quantifier Elimination and Cylindrical Algebraic Decomposition*, ed. by B.F. Caviness, J.R. Johnson (Springer, Berlin, Heidelberg 1998) pp. 8–23

Avidbots Ex1016
Page 207 of 2154

Multimedia Contents

8. Motion Control

Wan Kyun Chung, Li-Chen Fu, Torsten Kröger

This chapter will focus on the motion control of robotic rigid manipulators. In other words, this chapter does not treat the motion control of mobile robots, flexible manipulators, and manipulators with elastic joints. The main challenge in the motion control problem of rigid manipulators is the complexity of their dynamics and uncertainties. The former results from nonlinearity and coupling in the robot manipulators. The latter is twofold: structured and unstructured. Structured uncertainty means imprecise knowledge of the dynamic parameters and will be touched upon in this chapter, whereas unstructured uncertainty results from joint and link flexibility, actuator dynamics, friction, sensor noise, and unknown environment dynamics, and will be treated in other chapters.

In this chapter, we begin with an introduction to motion control of robot manipulators from a fundamental viewpoint, followed by a survey and brief review of the relevant advanced materials. Specifically, the dynamic model and useful properties of robot manipulators are recalled in Sect. 8.1. The joint and operational space control approaches, two different viewpoints on control of robot manipulators, are compared in Sect. 8.2. Independent joint control and proportional–integral–derivative (PID) control, widely adopted in the field of industrial robots, are presented in Sects. 8.3 and 8.4, respectively. Tracking control, based on feedback linearization, is introduced in Sect. 8.5. The computed-torque control and its variants are described in Sect. 8.6. Adaptive control is introduced in Sect. 8.7 to solve the problem of structural uncertainty, whereas the optimality and robustness issues are covered in Sect. 8.8. To compute suitable set point signals as input values for these motion controllers, Sect. 8.9 introduces reference trajectory planning concepts. Since most controllers of robot manipulators are implemented by using microprocessors, the issues of digital implementation are discussed in Sect. 8.10. Finally, learning control, one popular approach to intelligent control, is illustrated in Sect. 8.11.

8.1 **Introduction to Motion Control** 164
8.1.1 Dynamical Model 164
8.1.2 Control Tasks 165
8.1.3 Summary 165
8.2 **Joint Space Versus Operational Space Control** 166
8.2.1 Joint Space Control 166
8.2.2 Operational Space Control 166
8.3 **Independent-Joint Control** 167
8.3.1 Controller Design Based on the Single-Joint Model 167
8.3.2 Controller Design Based on the Multijoint Model 169
8.3.3 Summary 169
8.4 **PID Control** 169
8.4.1 PD Control for Regulation 169
8.4.2 PID Control for Regulation 170
8.4.3 PID Gain Tuning 170
8.4.4 Automatic Tuning 171
8.5 **Tracking Control** 172
8.5.1 Inverse Dynamics Control 172
8.5.2 Feedback Linearization 172
8.5.3 Passivity-Based Control 173
8.5.4 Summary 174
8.6 **Computed-Torque Control** 174
8.6.1 Computed-Torque Control 174
8.6.2 Computed-Torque-Like Control 175
8.6.3 Summary 177
8.7 **Adaptive Control** 177
8.7.1 Adaptive Computed-Torque Control 177
8.7.2 Adaptive Inertia-Related Control 178
8.7.3 Adaptive Control Based on Passivity 179

Avidbots Ex1016
Page 208 of 2154

8.7.4 Adaptive Control with Desired Compensation 180
8.7.5 Summary 180
8.8 **Optimal and Robust Control** 181
8.8.1 Quadratic Optimal Control 181
8.8.2 Nonlinear $\mathcal{H}_\infty$ Control 181
8.8.3 Passivity-Based Design of Nonlinear $\mathcal{H}_\infty$ Control 182
8.8.4 A Solution to Inverse Nonlinear $\mathcal{H}_\infty$ Control 183
8.9 **Trajectory Generation and Planning** 183
8.9.1 Geometric Paths and Trajectories 183
8.9.2 Joint Space and Operational Space Trajectories 184
8.9.3 Trajectory Representations 184
8.9.4 Trajectory Planning Algorithms ... 186
8.10 **Digital Implementation** 187
8.10.1 *Z*-Transform for Motion Control Implementation 188
8.10.2 Digital Control for Coding 189
8.11 **Learning Control** 190
8.11.1 Pure P-Type Learning Control 190
8.11.2 P-Type Learning Control with a Forgetting Factor 190
8.11.3 Summary 191
Video-References 191
References 191

8.1 Introduction to Motion Control

The dynamical model of robot manipulators will be recalled in this section. Furthermore, important properties of this dynamical model, which is useful in controller design, will then be addressed. Finally, different control tasks of the robot manipulators will be defined.

8.1.1 Dynamical Model

For motion control, the dynamical model of rigid robot manipulators is conveniently described by Lagrange dynamics. Let the robot manipulator have n links and let the $(n\times 1)$-vector $\boldsymbol{q}$ of joint variables be $\boldsymbol{q} = [q_1, \ldots, q_n]^{\mathrm{T}}$. The dynamic model of the robot manipulator is then described by Lagrange's equation [8.1–6]

$$\mathbf{H}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \mathbf{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{\tau}_{\mathrm{g}}(\boldsymbol{q}) = \boldsymbol{\tau} \ , \tag{8.1}$$

where $\mathbf{H}(\boldsymbol{q})$ is the $(n\times n)$ inertia matrix, $\mathbf{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}}$ is the $(n\times 1)$-vector of Coriolis and centrifugal forces, $\boldsymbol{\tau}_{\mathrm{g}}(\boldsymbol{q})$ is the $(n\times 1)$-vector of gravity force, and $\boldsymbol{\tau}$ is the $(n\times 1)$-vector of joint control inputs to be designed. Friction and disturbance input have been neglected here.

Remark 8.1
Other contributions to the dynamic description of the robot manipulators may include the dynamics of the actuators, joint and link flexibility, friction, noise, and disturbances. Without loss of generality, the case of the rigid robot manipulators is stressed here.

The control schemes that we will introduce in this chapter are based on some important properties of the dynamical model of robot manipulators. Before giving a detailed introduction to these different schemes, let us first give a list of those properties.

Property 8.1
The inertia matrix is a symmetric positive-definite matrix, which can be expressed

$$\lambda_{\mathrm{h}}\mathbf{I}_n \le \mathbf{H}(\boldsymbol{q}) \le \lambda_{\mathrm{H}}\mathbf{I}_n \ , \tag{8.2}$$

where λ_{h} and λ_{H} denote positive constants.

Property 8.2
The matrix $\mathbf{N}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \dot{\mathbf{H}}(\boldsymbol{q}) - 2\mathbf{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ is skew-symmetric for a particular choice of $\mathbf{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ (which is always possible), i. e.,

$$\boldsymbol{z}^{\mathrm{T}}\mathbf{N}(\boldsymbol{q}, \dot{\boldsymbol{q}})\boldsymbol{z} = 0 \tag{8.3}$$

for any $(n\times 1)$-vector $\boldsymbol{z}$.

Property 8.3
The $(n\times n)$-matrix $\mathbf{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ satisfies

$$\| \mathbf{C}(\boldsymbol{q}, \dot{\boldsymbol{q}}) \| \le c_{\mathrm{o}} \| \dot{\boldsymbol{q}} \| \tag{8.4}$$

for some bounded constant c_{o}.

Property 8.4
The gravity force/torque vector satisfies

$$\| \boldsymbol{\tau}_{\mathrm{g}}(\boldsymbol{q}) \| \le g_{\mathrm{o}} \tag{8.5}$$

for some bounded constant g_{o}.

Property 8.5
The equation of motion is linear in the inertia parameters. In other words, there is a $(r\times 1)$ constant

Avidbots Ex1016
Page 209 of 2154

vector $\boldsymbol{a}$ and an $(n \times r)$ regressor matrix $\mathbf{Y}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}})$ such that

$$\mathbf{H}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \mathbf{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{\tau}_{\mathrm{g}}(\boldsymbol{q}) = \mathbf{Y}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}})\boldsymbol{a} \,. \tag{8.6}$$

The vector $\boldsymbol{a}$ is comprised of link masses, moments of inertia, and the link, in various combinations.

Property 8.6

The mapping $\boldsymbol{\tau} \to \dot{\boldsymbol{q}}$ is passive; i. e., there exists $\alpha \geq 0$ such that

$$\int_0^t \dot{\boldsymbol{q}}^{\mathrm{T}}(\beta)\boldsymbol{\tau}(\beta)\,\mathrm{d}\beta \geq -\alpha \,, \quad \forall t < \infty \,. \tag{8.7}$$

Remarks 8.1

- Properties 8.3 and 8.4 are very useful since they allow us to establish upper bounds on the nonlinear terms in the dynamical model. As we will see further, several control schemes require knowledge of such upper bounds.
- In Property 8.5, the parameter vector $\boldsymbol{a}$ is comprised of several variables in various combinations. The dimensionality of the parameter space is not unique, and the search over the parameter space is an important problem.
- In this section, we assume that *the robot manipulator is fully actuated* and this indicates that there is an independent control input for each degree of freedom (DOF). In contrast, the robot manipulators with joint or link flexibility are no longer fully actuated and the control problems are more difficult in general.

8.1.2 Control Tasks

It is instructive for comparative purposes to classify control objectives into the following two classes:

- *Trajectory tracking* is aimed at following a time-varying joint reference trajectory specified within the manipulator workspace. In general, this desired trajectory is assumed to comply with the actuators' capacity. In other words, the joint velocity and acceleration associated with the desired trajectory should not violate, respectively, the velocity and acceleration limit of the manipulator. In practice, the capacity of actuators is set by torque limits, which result in bounds on the acceleration that are complex and state dependent.
- *Regulation* sometimes is also called point-to-point control. A fixed configuration in the joint space is specified; the objective is to bring to and keep the joint variable at the desired position in spite of torque disturbances and independently of the initial conditions. The behavior of transients and overshooting, are in general, not guaranteed.

The selection of the controller may depend on the type of task to be performed. For example, tasks only requiring the manipulator to move from one position to another without requiring significant precision during the motion between these two points can be solved by regulators, whereas such as welding, painting, and so on, require tracking controllers.

Remarks 8.2

- The regulation problem may be seen as a special case of the tracking problem (for which the desired joint velocity and acceleration are zero).
- The task specification above is given in the joint space and results in joint space control, which is the main content of this chapter. Sometimes, the task specification of the robot manipulators in terms of the desired trajectory of the end-effector (e.g., control with eye-in-hand) is carried out in the task space and gives rise to the operational space control, which will be introduced in Sect. 8.2.

8.1.3 Summary

In this section, we introduced the dynamical model of the robot manipulators and important properties of this dynamical model. Finally, we defined different control tasks of the robot manipulators.

Avidbots Ex1016
Page 210 of 2154

8.2 Joint Space Versus Operational Space Control

In a motion control problem, the manipulator moves to a position to pick up an object, transports that object to another location, and deposits it. Such a task is an integral part of any higher-level manipulation tasks such as painting or spot-welding.

Tasks are usually specified in the task space in terms of a desired trajectory of the end-effector, while control actions are performed in the joint space to achieve the desired goals. This fact naturally leads to two kinds of general control methods, namely *joint space control* and *operational space control (task space control)* schemes.

8.2.1 Joint Space Control

The main goal of the joint space control is to design a feedback controller such that the joint coordinates $\boldsymbol{q}(t) \in R^n$ track the desired motion $\boldsymbol{q}_{\mathrm{d}}(t)$ as closely as possible. To this end, consider the equations of motion (8.1) of an n-DOF manipulator expressed in the joint space [8.2, 4]. In this case, the control of robot manipulators is naturally achieved in the joint space, since the control inputs are the joint torques. Nevertheless, the user specifies a motion in terms of end-effector coordinates, and thus it is necessary to understand the following strategy.

Figure 8.1 shows the basic outline of the joint space control methods. Firstly, the desired motion, which is described in terms of end-effector coordinates, is converted to a corresponding joint trajectory using the inverse kinematics of the manipulator. Then the feedback controller determines the joint torque necessary to move the manipulator along the desired trajectory specified in joint coordinates starting from measurements of the current joint states [8.1, 4, 7, 8].

Since it is always assumed that the desired task is given in terms of the time sequence of the joint motion, joint space control schemes are quite adequate in situations where manipulator tasks can be accurately preplanned and little or no online trajectory adjustments are necessary [8.1, 4, 7, 9]. Typically, inverse kinematics is performed for some intermediate task points, and the joint trajectory is interpolated using the intermediate joint solutions. Although the command trajectory consists of straight-line motions in end-effector coordinates between interpolation points, the resulting joint motion consists of curvilinear segments that match the desired end-effector trajectory at the interpolation points.

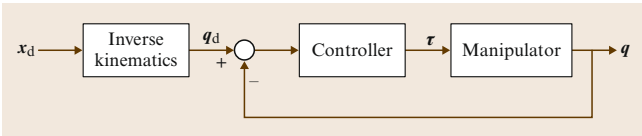


Fig. 8.1 Generic concept of joint space control

In fact, the joint space control includes simple proportional–derivative (PD) control, PID control, inverse dynamic control, Lyapunov-based control, and passivity-based control, as explained in the following sections.

8.2.2 Operational Space Control

In more complicated and less certain environments, end-effector motion may be subject to online modifications in order to accommodate unexpected events or to respond to sensor inputs. There are a variety of tasks in manufacturing where these type of control problem arise. In particular, it is essential when controlling the interaction between the manipulator and environment is of concern.

Since the desired task is often specified in the operational space and requires precise control of the end-effector motion, joint space control schemes are not suitable in these situations. This motivated a different approach, which can develop control schemes directly based on the dynamics expressed in the operational space [8.10, 11].

Let us suppose that the Jacobian matrix, denoted by $\mathbf{J}(\boldsymbol{q}) \in R^{n\times n}$, transforms the joint velocity ($\dot{\boldsymbol{q}} \in R^n$) to the task velocity ($\dot{\boldsymbol{x}} \in R^n$) according to

$$\dot{\boldsymbol{x}} = \mathbf{J}(\boldsymbol{q})\dot{\boldsymbol{q}} \,. \tag{8.8}$$

Furthermore, assume that it is invertible. Then, the operational space dynamics is expressed as follows

$$\boldsymbol{f}_{\mathrm{c}} = \boldsymbol{\Lambda}(\boldsymbol{q})\ddot{\boldsymbol{x}} + \boldsymbol{\Gamma}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{x}} + \boldsymbol{\eta}(\boldsymbol{q}) \,, \tag{8.9}$$

where $\boldsymbol{f}_{\mathrm{c}} \in R^n$ denotes the command forces in the operational space; the *pseudo-inertia matrix* is defined by

$$\boldsymbol{\Lambda}(\boldsymbol{q}) = \mathbf{J}^{-\mathrm{T}}(\boldsymbol{q})\mathbf{H}(\boldsymbol{q})\mathbf{J}^{-1}(\boldsymbol{q}) \,, \tag{8.10}$$

and $\boldsymbol{\Gamma}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ and $\boldsymbol{\eta}(\boldsymbol{q})$ are given by

$$\begin{aligned}\boldsymbol{\Gamma}(\boldsymbol{q}, \dot{\boldsymbol{q}}) &= \mathbf{J}^{-\mathrm{T}}(\boldsymbol{q})\mathbf{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\mathbf{J}^{-1}(\boldsymbol{q}) \\ &\quad - \boldsymbol{\Lambda}(\boldsymbol{q})\mathbf{J}(\boldsymbol{q})\mathbf{J}^{-1}(\boldsymbol{q}) \,, \\ \boldsymbol{\eta}(\boldsymbol{q}) &= \mathbf{J}^{-\mathrm{T}}(\boldsymbol{q})\boldsymbol{\tau}_{\mathrm{g}}(\boldsymbol{q}) \,.\end{aligned}$$

The task space variables are usually reconstructed from the joint space variables, via the kinematic mappings. In fact, it is quite rare to have sensors to directly measure end-effector positions and velocities. Also, it

Avidbots Ex1016
Page 211 of 2154

is worth remarking that an analytical Jacobian is utilized since the control schemes operate directly on task space quantities, i. e., the end-effector position and orientation.

The main goal of the operational space control is to design a feedback controller that allows execution of an end-effector motion $\boldsymbol{x}(t) \in R^n$ that tracks the desired end-effector motion $\boldsymbol{x}_d(t)$ as closely as possible. To this end, consider the equations of motion (8.9) of the manipulator expressed in the operational space. For this case, Fig. 8.2 shows a schematic diagram of the operational space control methods. There are several advantages to such an approach because operational space controllers employ a feedback loop that directly minimizes task errors. Inverse kinematics need not be calculated explicitly, since the control algorithm embeds the velocity-level forward kinematics (8.8), as shown in the figure. Now, motion between points can be a straight-line segment in the task space.

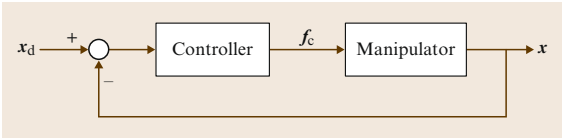


Fig. 8.2 Basic concept of operational space control

8.3 Independent-Joint Control

By independent-joint control (i. e., decentralized control), we mean that the control inputs of each joint only depends on the measurement of the corresponding joint displacement and velocity. Due to its simple structure, this kind of control schemes offers many advantages. For example, by using independent-joint control, communication among different joints is saved. Moreover, since the computational load of controllers may be reduced, only low-cost hardware is required in actual implementations. Finally, independent-joint control has the feature of scalability, since the controllers on all the joints have the same formulation. In this section, two kinds of design of independent-joint control will be introduced: one focused on the dynamical model of each joint (i. e., based on the single-joint model) and the other based on the analysis of the overall dynamical model (i. e., the multijoint model) of robot manipulators.

8.3.1 Controller Design Based on the Single-Joint Model

The simplest independent-joint control strategy is to control each joint axis as a single-input single-output (SISO) system. Coupling effects among joints due to varying configuration during motion are treated as disturbance inputs. Without loss of generality, the actuator is taken as a rotary electric direct-current (DC) motor. Hence, the block diagram of the control scheme of joint i can be represented in the domain of the complex variables as shown in Fig. 8.3. In this scheme, θ is the angular variable of the motor, J is the effective inertia viewed from the motor side, R_a is the armature resistance (auto-inductance being neglected), and k_t and k_v are, respectively, the torque and motor constants. Furthermore, G_v denotes the voltage gain of the power amplifier so that the reference input is the input voltage V_c of the amplifier instead of the armature voltage V_a. It has also been assumed that $F_m \ll k_v k_t / R_a$, i. e., the mechanical (viscous) friction coefficient has been neglected with respect to the electrical coefficient. Now the input–output transfer function of the motor can be written as

$$M(s) = \frac{k_m}{s(1 + sT_m)} , \tag{8.11}$$

where $k_m = G_v / k_v$ and $T_m = R_a J / k_v k_t$ are, respectively, the voltage-to-velocity gain and the time constant of the motor.

To guide selection of the control structure, start by noticing that an effective rejection of the disturbance d on the output θ is ensured by

1. A large value of the amplifier before the point of intervention of the disturbance
2. The presence of an integral action in the controller so as to cancel the effect of the gravitational component on the output at the steady state (i. e., constant θ).

In this case, as shown in Fig. 8.4, the types of control action with position and velocity feedback are

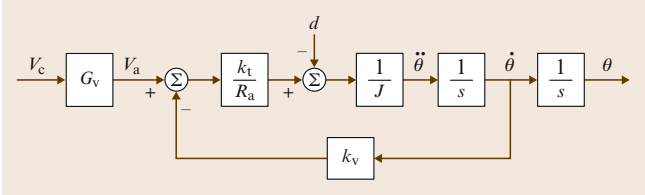


Fig. 8.3 Block scheme of a joint-driven system (after [8.4])

Avidbots Ex1016
Page 212 of 2154

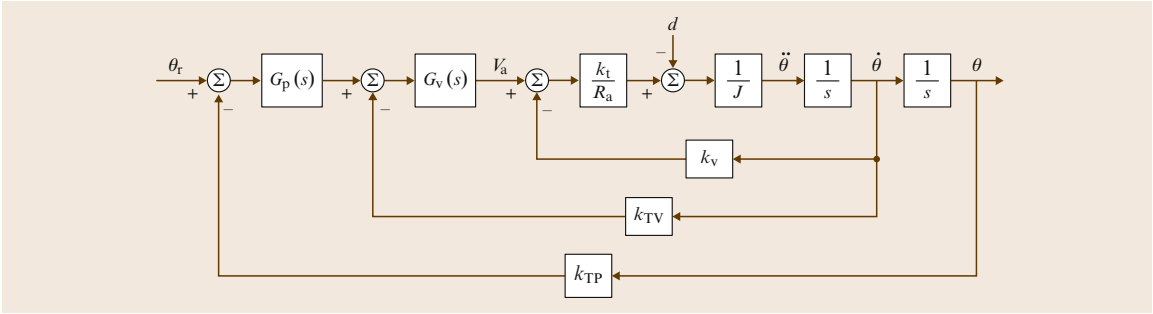


Fig. 8.4 Block scheme of position and velocity feedback (after [8.4])

characterized by [8.4]

$$G_p(s) = K_P\,, \quad G_v(s) = K_V \frac{1 + sT_V}{s}\,, \tag{8.12}$$

where $G_p(s)$ and $G_v(s)$ correspond to the position and velocity control actions, respectively. It is worth noting that the inner control action $G_v(s)$ is in a form of propositional–integral (PI) control to yield zero error in the steady state under a constant disturbance d. Furthermore, k_{TP} and k_{TV} are both transducer constants, and the amplifier gain K_V has been embedded in the gain of the inner controller. From the scheme of Fig. 8.4, the transfer function of the forward path is

$$P(s) = \frac{k_m K_P K_V (1 + sT_V)}{s(1 + sT_m)}\,, \tag{8.13}$$

while that of the return path is

$$H(s) = k_{TP}\left(1 + s\frac{k_{TV}}{K_P k_{TP}}\right)\,. \tag{8.14}$$

The zero of the controller at $s = -1/T_V$ can be chosen so as to cancel the effects of the real pole of the motor at $s = -1/T_m$. Then, by setting $T_V = T_m$, the poles of the closed-loop system move on the root locus as a function of the loop gain, $k_m K_V k_{TV}$. By increasing the position feedback gain K_P, it is possible to confine the closed-loop poles to a region of the complex plane with large absolute real part. Then, the actual location can be established by a suitable choice of K_V.

The closed-loop input–output transfer function is

$$\frac{\Theta(s)}{\Theta_r(s)} = \frac{\dfrac{1}{k_{TP}}}{1 + \dfrac{sk_{TP}}{K_P k_{TP}} + \dfrac{s^2}{k_m K_P k_{TP} K_V}}\,, \tag{8.15}$$

which can be compared with the typical transfer function of a second-order system

$$W(s) = \frac{\dfrac{1}{k_{TP}}}{1 + \dfrac{2\zeta s}{\omega_n} + \dfrac{s^2}{\omega_n^2}}\,. \tag{8.16}$$

It can be recognized that, with a suitable choice of the gains, it is possible to obtain any value of natural frequency ω_n and damping ratio ζ. Hence, if ω_n and ζ are given as design specifications, the following relations can be found

$$K_V k_{TV} = \frac{2\zeta\omega_n}{k_m} \quad \text{and} \quad K_P k_{TP} K_V = \frac{\omega_n^2}{k_m}\,. \tag{8.17}$$

For given transducer constants k_{TP} and k_{TV}, K_V and K_P will be chosen to satisfy the two equations above, respectively. On the other hand, the closed-loop disturbance/output function is

$$\frac{\Theta(s)}{D(s)} = \frac{\dfrac{sR_a}{k_t K_P K_{TP} K_V (1 + sT_m)}}{1 + \dfrac{sk_{TV}}{K_P k_{TP}} + \dfrac{s^2}{k_m K_P k_{TP} K_V}}\,, \tag{8.18}$$

which shows that the disturbance rejection factor is $X_R(s) = K_P k_{TP} K_V$ and is fixed, provided that K_V and K_P have been chosen via the approach above. Concerning the disturbance dynamics, the zero at the origin introduced by the PI, a real pole at $s = -1/T_m$, and the pair of complex poles with real part $-\zeta\omega_n$ should be kept in mind. In this case, an estimate T_R of the output recovery time needed by the control system to recover from the effect of a disturbance on the joint position can be evaluated by analyzing models of the transfer function above. Such an estimate can reasonably be expressed as $T_R = \max\{T_m, 1/\zeta\omega\}$.

Avidbots Ex1016
Page 213 of 2154

8.3.2 Controller Design Based on the Multijoint Model

In recent years, independent-joint control schemes based on the complete dynamic model of the robot manipulators (i. e., a multijoint model) have been proposed. For example, following the approach of computed-torque-like control, [8.12] dealt with the regulation task for horizontal motions, and [8.13] and [8.14] handled the tracking task for arbitrary smooth trajectories. Since the overall dynamic model is considered, the effects of coupling among joints are handled. These schemes will be introduced in detail in Sect. 8.6.

8.3.3 Summary

In this section, we have presented two independent-joint control schemes: one is based on the single-joint model and the other is based on the multijoint model. The former focuses on the dynamics of a single joint and regards the interaction among joints as a disturbance. This control scheme is simple but may not be suitable for high-speed tracking. Hence, we introduce the latter, which considers the overall dynamical model of robot manipulators such that the interaction among joints can be handled.

Further Reading

There are different types of feedback applied in the independent-joint control based on the single-joint model (such as pure position feedback or position, velocity, and acceleration feedback). A complete discussion is given in [8.4]. When the joint control servos are required to track reference trajectories with high speeds and accelerations, the tracking capabilities of the above schemes are inevitably degraded. A possible remedy is to adopt decentralized feedforward compensation to reduce tracking errors [8.4, 5].

8.4 PID Control

Traditionally, control design in robot manipulators can be understood as the simple fact of tuning of a PD or PID compensator at the level of each motor driving the manipulator joints [8.1]. Fundamentally, a PD controller is a position and velocity feedback that has good closed-loop properties when applied to a double integrator system.

The PID control has a long history since *Ziegler* and *Nichols*' PID tuning rules were published in 1942 [8.15]. Actually, the strong point of PID control lies in its *simplicity* and clear physical meaning. Simple control is preferable to complex control, at least in industry, if the performance enhancement obtained by using complex control is not significant enough. The physical meanings of PID control [8.16] are as follows:

- *P-control* means the present effort making a present state into desired state
- *I-control* means the accumulated effort using the experience information of previous states
- *D-control* means the predictive effort reflecting the information about trends in future states.

8.4.1 PD Control for Regulation

A simple design method for manipulator control is to utilize a linear control scheme based on the linearization of the system about an operating point. An example of this method is a PD control with a gravity compensation scheme [8.17, 18]. Gravity compensation acts as a bias correction, compensating only for the amount of forces that create overshooting and an asymmetric transient behavior. Formally, it has the following form

$$\boldsymbol{\tau} = \mathbf{K}_\mathrm{P}(\boldsymbol{q}_\mathrm{d} - \boldsymbol{q}) - \mathbf{K}_\mathrm{V}\dot{\boldsymbol{q}} + \boldsymbol{\tau}_\mathrm{g}(\boldsymbol{q}) \,, \tag{8.19}$$

where $\mathbf{K}_\mathrm{P}$ and $\mathbf{K}_\mathrm{V} \in R^{n\times n}$ are positive-definite gain matrices. This controller is very useful for set-point regulation, i. e., $\boldsymbol{q}_\mathrm{d}$ = constant [8.7, 18]. When this controller is applied to (8.1), the closed-loop equation becomes

$$\mathbf{H}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \mathbf{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \mathbf{K}_\mathrm{V}\dot{\boldsymbol{q}} - \mathbf{K}_\mathrm{P}\boldsymbol{e}_\mathrm{q} = \mathbf{0} \,, \tag{8.20}$$

where $\boldsymbol{e}_\mathrm{q} = \boldsymbol{q}_\mathrm{d} - \boldsymbol{q}$, and the equilibrium point is $\boldsymbol{y} = [\boldsymbol{e}_\mathrm{q}^\mathrm{T}, \dot{\boldsymbol{q}}^\mathrm{T}]^\mathrm{T} = \mathbf{0}$. Now, the stability achieved by PD control with gravity compensation can be analyzed according to the closed-loop dynamic (8.20). Consider the positive-definite function

$$V = \frac{1}{2}\dot{\boldsymbol{q}}^\mathrm{T}\mathbf{H}(\boldsymbol{q})\dot{\boldsymbol{q}} + \frac{1}{2}\boldsymbol{e}_\mathrm{q}^\mathrm{T}\mathbf{K}_\mathrm{V}\boldsymbol{e}_\mathrm{q} \,.$$

Then, the derivative of function becomes negative semidefinite for any value of $\dot{\boldsymbol{q}}$ by using Property 8.2 in Sect. 8.1, i. e.,

$$\dot{V} = -\dot{\boldsymbol{q}}^\mathrm{T}\mathbf{K}_\mathrm{V}\dot{\boldsymbol{q}} \le -\lambda_{\min}(\mathbf{K}_\mathrm{V})\|\dot{\boldsymbol{q}}\|^2 \,, \tag{8.21}$$

where $\lambda_{\min}(\mathbf{K}_\mathrm{V})$ means the smallest eigenvalue of $\mathbf{K}_\mathrm{V}$. By invoking the Lyapunov stability theory and LaSalle's theorem [8.1], it can be shown that the regulation error will converge asymptotically to zero, while

their high-order derivatives remain bounded. This controller requires knowledge of the gravity components (structure and parameters), though it is simple.

Now, consider simple PD control without gravity compensation

$$\boldsymbol{\tau} = \mathbf{K}_\mathrm{P}(\boldsymbol{q}_\mathrm{d} - \boldsymbol{q}) - \mathbf{K}_\mathrm{V}\dot{\boldsymbol{q}} \,, \tag{8.22}$$

then the closed-loop dynamic equation becomes

$$\mathbf{H}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \mathbf{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{\tau}_\mathrm{g}(\boldsymbol{q}) + \mathbf{K}_\mathrm{V}\dot{\boldsymbol{q}} - \mathbf{K}_\mathrm{P}\boldsymbol{e}_\mathrm{q} = \mathbf{0} \,. \tag{8.23}$$

Consider the positive definite function

$$V = \frac{1}{2}\dot{\boldsymbol{q}}^\mathrm{T}\mathbf{H}(\boldsymbol{q})\dot{\boldsymbol{q}} + \frac{1}{2}\boldsymbol{e}_\mathrm{q}^\mathrm{T}\mathbf{K}_\mathrm{V}\boldsymbol{e}_\mathrm{q} + U(\boldsymbol{q}) + U_0 \,,$$

where $U(\boldsymbol{q})$ denotes the potential energy with the relation of $\partial U(\boldsymbol{q})/\partial \boldsymbol{q} = \boldsymbol{\tau}_\mathrm{g}(\boldsymbol{q})$ and U_0 is a suitable constant. Taking time derivative of V along the closed-loop dynamics (8.23) gives the same result (8.21) with previous one using gravity compensation. In this case, the control system must be stable in the sense of Lyapunov, but it can not conclude that the regulation error will converge to zero by LaSalle's theorem [8.1]. Actually, the system precision (the size of the regulation error vector) will depend on the size of the gain matrix $\mathbf{K}_\mathrm{P}$ in the following form

$$\|\boldsymbol{e}_\mathrm{q}\| \le \|\mathbf{K}_\mathrm{P}^{-1}\| g_0 \,, \tag{8.24}$$

where g_0 is that in Property 8.4 in Sect. 8.1. Hence, the regulation error can be arbitrarily reduced by increasing $\mathbf{K}_\mathrm{P}$; nevertheless, measurement noise and other unmodeled dynamics, such as actuator friction, will limit the use of high gains in practice.

8.4.2 PID Control for Regulation

An integral action may be added to the previous PD control in order to deal with gravity forces, which to some extent can be considered as a constant disturbance (from the local point of view). The PID regulation controller can be written in the following general form

$$\boldsymbol{\tau} = \mathbf{K}_\mathrm{P}(\boldsymbol{q}_\mathrm{d} - \boldsymbol{q}) + \mathbf{K}_\mathrm{I}\int f(\boldsymbol{q}_\mathrm{d} - \boldsymbol{q})\mathrm{d}t - \mathbf{K}_\mathrm{V}\dot{\boldsymbol{q}} \,,$$

where $\mathbf{K}_\mathrm{I} \in R^{n\times n}$ is a positive-definite gain matrix, and:

- If $f(\boldsymbol{q}_\mathrm{d} - \boldsymbol{q}) = \boldsymbol{q}_\mathrm{d} - \boldsymbol{q}$, we have PID control.
- If $\mathbf{K}_\mathrm{I}\int(-\dot{\boldsymbol{q}})\mathrm{d}t$ is added, we have PI^{2}D control.
- If $f(\cdot) = \tanh(\cdot)$, we have PD + nonlinear integral control.

Global asymptotic stability (GAS) by PID control was proved in [8.12] for robotic motion control system including external disturbances, such as Coulomb friction. Also, *Tomei* proved the GAS of PD control in [8.19] by using an adaptation for gravity term. On the other hand, *Ortega* et al. showed in [8.20] that PI^{2}D control could yield semiglobal asymptotic stability (SGAS) in the presence of gravity and bounded external disturbances. Also, *Angeli* proved in [8.21] that PD control could achieve the input-output-to-state stability (IOSS) for robotic systems. Also, *Ramirez* et al. proved the SGAS (with some conditions) for PID gains in [8.22]. Also, *Kelly* proved in [8.23] that PD plus nonlinear integral control could achieve GAS under gravity.

Actually, a large integral action in PID control can cause instability of the motion control system. In order to avoid this, the integral gain should be bounded by an upper limit of the following form [8.1]

$$\frac{k_\mathrm{P}k_\mathrm{V}}{\lambda_\mathrm{H}^2} > k_\mathrm{I} \,,$$

where λ_H is that in Property 8.1 in Sect. 8.1, $\mathbf{K}_\mathrm{P} = k_\mathrm{P}\mathbf{I}$, $\mathbf{K}_\mathrm{I} = k_\mathrm{I}\mathbf{I}$, and $\mathbf{K}_\mathrm{V} = k_\mathrm{V}\mathbf{I}$. This relation gives the guidelines for gain selection implicitly. Also, PID control has generated a great variety of PID control plus something, e.g., PID plus friction compensator, PID plus gravity compensator, PID plus disturbance observer.

8.4.3 PID Gain Tuning

The PID control can be utilized for trajectory tracking as well as set-point regulation. True tracking control will be treated after Sect. 8.5. In this section, the simple but useful PID gain tuning method will be introduced for practical use. The general PID controller can be written in the following general form

$$\boldsymbol{\tau} = \mathbf{K}_\mathrm{V}\dot{\boldsymbol{e}}_\mathrm{q} + \mathbf{K}_\mathrm{P}\boldsymbol{e}_\mathrm{q} + \mathbf{K}_\mathrm{I}\int \boldsymbol{e}_\mathrm{q}\mathrm{d}t \,,$$

or, in another form,

$$\boldsymbol{\tau} = \left(\mathbf{K} + \frac{1}{\gamma^2}\mathbf{I}\right)\left(\dot{\boldsymbol{e}}_\mathrm{q} + \mathbf{K}_\mathrm{P}\boldsymbol{e}_\mathrm{q} + \mathbf{K}_\mathrm{I}\int \boldsymbol{e}_\mathrm{q}\mathrm{d}t\right) \,. \tag{8.25}$$

In a fundamental stability analysis of tracking control systems, *Qu* and *Dorsey* proved in [8.24] that PD control could satisfy uniform ultimate boundedness (UUB). Also, *Berghuis* and *Nijmeijer* proposed output feedback PD control, which satisfies semiglobal uniform ultimate boundedness (SGUUB) in [8.25] under gravity and a bounded disturbance. Recently, *Choi* et al. suggested inverse optimal PID control [8.26], assuring extended disturbance input-to-state stability (ISS).

Avidbots Ex1016
Page 215 of 2154

Actually, if a PID controller (8.25) is repeatedly applied to the same set point or desired trajectory, then the maximum error will be proportional to the gains in the following form

$$\max_{0\le t\le t_f} \|\boldsymbol{e}_q(t)\| \propto \frac{\gamma^2}{\sqrt{2k\gamma^2+1}}\,, \tag{8.26}$$

where t_f denotes the final execution time of a given task and $\mathbf{K} = k\mathbf{I}$. This relation can be utilized to tune the gain of a PID controller and is referred to as the compound tuning rule [8.16]. The compound tuning rule implicitly includes simple tuning rules as follows (γ tuning method):

- Square tuning: $\max \|\boldsymbol{e}_q\| \propto \gamma^2$, for a small k
- Linear tuning: $\max \|\boldsymbol{e}_q\| \propto \gamma$, for a large k.

For example, suppose we select positive constant diagonal matrices $\mathbf{K}_P = k_P\mathbf{I}, \mathbf{K}_I = k_I\mathbf{I}$, while satisfying $k_P^2 > 2k_I$. For small k, the maximum error will be reduced by $\frac{1}{4}$ according to the square tuning rule, if we reduce the value γ by $\frac{1}{2}$. For large k, the maximum error will be proportionally reduced as γ according to the linear tuning rule. This means that we can tune the PID controller using only one parameter γ when the other gain parameters are fixed [8.16] (VIDEO 25). Although these rules are very useful in tuning the control performance, they can be utilized only for repetitive experiments for the same set point or desired trajectory because the tuning rules consist of proportional relations.

8.4.4 Automatic Tuning

For simplicity, define the composite error to be

$$\boldsymbol{s}(t) = \dot{\boldsymbol{e}}_q + \mathbf{K}_P\boldsymbol{e}_q + \mathbf{K}_I\int \boldsymbol{e}_q dt\,.$$

Now simple auto-tuning PID control is suggested by choosing one tuning parameter $\mathbf{K}$ as shown in the following control form

$$\boldsymbol{\tau} = \left(\widehat{K}(t) + \frac{1}{\gamma^2}\mathbf{I}\right)\boldsymbol{s}(t)\,,$$

and its automatic tuning rule as follow

$$\frac{d\widehat{K}_i}{dt} = \Gamma_i s_i^2(t), \quad \text{for } i = 1,\cdots,n\,, \tag{8.27}$$

where Γ_i implies an update gain parameter for i-th control joint.

For practical use of the PID control, a target performance denoted by Ω is specified in advance

$$\sup_{0\le t\le t_f} |\boldsymbol{s}(t)| = \Omega\,,$$

in order to maintain the composite error within the region Ω. Moreover, since the auto-tuning rule has the decentralized type (8.27), we suggest the decentralized criterion for auto-tuning as follows

$$|s_i| > \frac{\Omega}{\sqrt{2n}}\,, \tag{8.28}$$

where n is the number of the joint coordinates. As soon as the composite error arrives at the tuning region (8.28), the auto-tuning rule is implemented to assist the achievement of target performance. On the contrary, if the composite error stays in non-tuning region, namely, $|s_i| \le \Omega/\sqrt{2n}$, then the auto-tuning process stops. For this case, we expect that the gain $\widehat{K}$ updated by an auto-tuning rule (8.27) would be larger than the matrix $\mathbf{K}_\Omega$ able to achieve the target performance Ω. As a matter of fact, the auto-tuning rule plays a nonlinear damping role in the auto-tuning region.

Matlab Example (Multimedia)

Simple automatic tuning example for one-link manipulator control system is shown in the multimedia source to help readers' understanding.

Further Reading

The PID-type controllers were designed to solve the regulation control problem. They have the advantage of requiring knowledge of neither the model structure nor the model parameters. Also, the stability achieved by PID-type controllers was presented in this section. A range of books and papers [8.1, 15, 16, 22, 27, 28] are available to the robotics audience, detailing the individual tuning methods used in PID control and their concrete proofs.

Avidbots Ex1016
Page 216 of 2154

8.5 Tracking Control

While independent PID controls are adequate in most set-point regulation problems, there are many tasks that require effective trajectory tracking capabilities such as plasma-welding, laser-cutting or high-speed operations in the presence of obstacles. In this case, employing local schemes requires moving slowly through a number of intermediate set points, thus considerably delaying the completion of the task. Therefore, to improve the trajectory tracking performance, controllers should take account of the manipulator dynamic model via a computed-torque-like technique.

The tracking control problem in the joint or task space consists of following a given time-varying trajectory $\boldsymbol{q}_\mathrm{d}(t)$ or $\boldsymbol{x}_\mathrm{d}(t)$ and its successive derivatives $\dot{\boldsymbol{q}}_\mathrm{d}(t)$ or $\dot{\boldsymbol{x}}_\mathrm{d}(t)$ and $\ddot{\boldsymbol{q}}_\mathrm{d}(t)$ or $\ddot{\boldsymbol{x}}_\mathrm{d}(t)$, which describe the desired velocity and acceleration, respectively. To obtain successful performance, significant effort has been devoted to the development of model-based control strategies [8.1, 2, 7]. Among the control approaches reported in the literature, typical methods include the inverse dynamics control, the feedback linearization technique, and the passivity-based control method.

8.5.1 Inverse Dynamics Control

Though the inverse dynamics control has a theoretical background, such as the theory of feedback linearization discussed later, its starting point is mechanical engineering intuition based on cancelling nonlinear terms and decoupling the dynamics of each link. Inverse dynamics control in joint space has the form

$$\boldsymbol{\tau} = \mathbf{H}(\boldsymbol{q})\boldsymbol{v} + \mathbf{C}(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{\tau}_\mathrm{g}(\boldsymbol{q}) \,, \tag{8.29}$$

which, applied to (8.1), yields a set of n decoupled linear systems, e.g., $\ddot{\boldsymbol{q}} = \boldsymbol{v}$, where $\boldsymbol{v}$ is an auxiliary control input to be designed. Typical choices for $\boldsymbol{v}$ are

$$\boldsymbol{v} = \ddot{\boldsymbol{q}}_\mathrm{d} + \mathbf{K}_\mathrm{V}(\dot{\boldsymbol{q}}_\mathrm{d} - \dot{\boldsymbol{q}}) + \mathbf{K}_\mathrm{P}(\boldsymbol{q}_\mathrm{d} - \boldsymbol{q}) \,, \tag{8.30}$$

or with an integral component

$$\boldsymbol{v} = \ddot{\boldsymbol{q}}_\mathrm{d} + \mathbf{K}_\mathrm{V}(\dot{\boldsymbol{q}}_\mathrm{d} - \dot{\boldsymbol{q}}) + \mathbf{K}_\mathrm{P}(\boldsymbol{q}_\mathrm{d} - \boldsymbol{q}) + \mathbf{K}_\mathrm{I}\int(\boldsymbol{q}_\mathrm{d} - \boldsymbol{q})\mathrm{d}t \,, \tag{8.31}$$

leading to the error dynamics equation

$$\ddot{\boldsymbol{e}}_\mathrm{q} + \mathbf{K}_\mathrm{V}\dot{\boldsymbol{e}}_\mathrm{q} + \mathbf{K}_\mathrm{P}\boldsymbol{e}_\mathrm{q} = \mathbf{0}$$

for an auxiliary control input (8.30), and

$$\boldsymbol{e}_\mathrm{q}^{(3)} + \mathbf{K}_\mathrm{V}\ddot{\boldsymbol{e}}_\mathrm{q} + \mathbf{K}_\mathrm{P}\dot{\boldsymbol{e}}_\mathrm{q} + \mathbf{K}_\mathrm{I}\boldsymbol{e}_\mathrm{q} = \mathbf{0} \,,$$

if an auxiliary control input (8.31) is used. Both error dynamics are exponentially stable by a suitable choice of the gain matrices $\mathbf{K}_\mathrm{V}$ and $\mathbf{K}_\mathrm{P}$ (and $\mathbf{K}_\mathrm{I}$).

Alternatively, inverse dynamics control can be described in the operational space. Consider the operational space dynamics (8.9). If the following inverse dynamics control is used in the operational space,

$$\boldsymbol{f}_\mathrm{c} = \boldsymbol{\Lambda}(\boldsymbol{q})(\ddot{\boldsymbol{x}}_\mathrm{d} + \mathbf{K}_\mathrm{V}\dot{\boldsymbol{e}}_x + \mathbf{K}_\mathrm{P}\boldsymbol{e}_x) + \boldsymbol{\Gamma}(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{x}} + \boldsymbol{\eta}(\boldsymbol{q}) \,,$$

where $\boldsymbol{e}_x = \boldsymbol{x}_\mathrm{d} - \boldsymbol{x}$, the resulting error dynamics is

$$\ddot{\boldsymbol{e}}_x + \mathbf{K}_\mathrm{V}\dot{\boldsymbol{e}}_x + \mathbf{K}_\mathrm{P}\boldsymbol{e}_x = \mathbf{0} \,, \tag{8.32}$$

and it is also exponentially stable. One apparent advantage of using this controller is that $\mathbf{K}_\mathrm{P}$ and $\mathbf{K}_\mathrm{V}$ can be selected with a clear physical meaning in operational space. However, as can be seen in (8.10), $\boldsymbol{\Lambda}(\boldsymbol{q})$ becomes very large when the robot approaches singular configurations [8.8]. This means that large forces in some direction are needed to move the arm.

8.5.2 Feedback Linearization

This approach generalizes the concept of inverse dynamics of rigid manipulators. The basic idea of feedback linearization is to construct a transformation as a so-called *inner-loop control*, which exactly linearizes the nonlinear system after a suitable state space change of coordinates. One can then design a second stage or *outer-loop control* in the new coordinates to satisfy the traditional control design specifications such as tracking, disturbance rejection, etc. [8.5, 29]. The full power of the feedback linearization scheme for manipulator control becomes apparent if one includes in the dynamic description of the manipulator the transmission dynamics, such as the elasticity resulting from shaft windup, gear elasticity, etc. [8.5].

In recent years, an impressive volume of literature has emerged in the area of differential-geometric methods for nonlinear systems. Most of the results in this area are intended to give abstract coordinate-free descriptions of various geometric properties of nonlinear systems and as such are difficult for the non-mathematician to follow. It is our intention here to give only the basic idea of the feedback linearization scheme and to introduce a simple version of this technique that finds an immediate application to the manipulator control problem. The reader is referred to [8.30] for a comprehensive treatment of the feedback linearization technique using differential-geometric methods.

Let us now develop a simple approach to the determination of linear state-space representations of the

Avidbots Ex1016
Page 217 of 2154

manipulator dynamics (8.1) by considering a general sort of output $\boldsymbol{\xi} \in R^p$

$$\boldsymbol{\xi} = \boldsymbol{h}(\boldsymbol{q}) + \boldsymbol{r}(t) \,, \tag{8.33}$$

where $\boldsymbol{h}(\boldsymbol{q})$ is a general predetermined function of the joint coordinate $\boldsymbol{q} \in R^n$ and $\boldsymbol{r}(t)$ is a general predetermined time function. The control objective will be to select the joint torque inputs $\boldsymbol{\tau}$ in order to make the output $\boldsymbol{\xi}(t)$ go to zero.

The choice of $\boldsymbol{h}(\boldsymbol{q})$ and $\boldsymbol{r}(t)$ is based on the control purpose. For example, if $\boldsymbol{h}(\boldsymbol{q}) = -\boldsymbol{q}$ and $\boldsymbol{r}(t) = \boldsymbol{q}_\mathrm{d}(t)$, the desired joint space trajectory we would like the manipulator to follow, then $\boldsymbol{\xi}(t) = \boldsymbol{q}_\mathrm{d}(t) - \boldsymbol{q}(t) \equiv \boldsymbol{e}_\mathrm{q}(t)$ is the joint space tracking error. Forcing $\boldsymbol{\xi}(t)$ to zero in this case would cause the joint variables $\boldsymbol{q}(t)$ to track their desired values $\boldsymbol{q}_\mathrm{d}(t)$, resulting in a manipulator trajectory-following problem. As another example, $\boldsymbol{\xi}(t)$ could represent the operational space tracking error $\boldsymbol{\xi}(t) = \boldsymbol{x}_\mathrm{d}(t) - \boldsymbol{x}(t) \equiv \boldsymbol{e}_x(t)$. Then, controlling $\boldsymbol{\xi}(t)$ to zero would result in trajectory following directly in operational space where the desired motion is usually specified.

To determine a linear state-variable model for manipulator controller design, let us simply differentiate the output $\boldsymbol{\xi}(t)$ twice to obtain

$$\dot{\boldsymbol{\xi}} = \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{q}} \dot{\boldsymbol{q}} + \dot{\boldsymbol{r}} = \mathbf{T}\dot{\boldsymbol{q}} + \dot{\boldsymbol{r}} \,, \tag{8.34}$$

$$\ddot{\boldsymbol{\xi}} = \mathbf{T}\ddot{\boldsymbol{q}} + \dot{\mathbf{T}}\dot{\boldsymbol{q}} + \ddot{\boldsymbol{r}} \,, \tag{8.35}$$

where we defined a $(p \times n)$ transformation matrix of the form

$$\mathbf{T}(\boldsymbol{q}) = \frac{\partial \boldsymbol{h}(\boldsymbol{q})}{\partial \boldsymbol{q}} = \left(\frac{\partial \boldsymbol{h}}{\partial q_1} \quad \frac{\partial \boldsymbol{h}}{\partial q_2} \quad \cdots \quad \frac{\partial \boldsymbol{h}}{\partial q_n} \right) . \tag{8.36}$$

Given the output $\boldsymbol{h}(\boldsymbol{q})$, it is straightforward to compute the transformation $\mathbf{T}(\boldsymbol{q})$ associated with $\boldsymbol{h}(\boldsymbol{q})$. In the special case where $\dot{\boldsymbol{\xi}}$ represents the operational space velocity error, then $\mathbf{T}(\boldsymbol{q})$ denotes the Jacobian matrix $\mathbf{J}(\boldsymbol{q})$.

According to (8.1),

$$\ddot{\boldsymbol{q}} = \mathbf{H}^{-1}(\boldsymbol{q})[\boldsymbol{\tau} - \boldsymbol{n}(\boldsymbol{q}, \dot{\boldsymbol{q}})] \,, \tag{8.37}$$

with the nonlinear terms represented by

$$\boldsymbol{n}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \mathbf{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{\tau}_\mathrm{g}(\boldsymbol{q}) \,. \tag{8.38}$$

Then (8.35) yields

$$\ddot{\boldsymbol{\xi}} = \ddot{\boldsymbol{r}} + \dot{\mathbf{T}}\dot{\boldsymbol{q}} + \mathbf{T}(\boldsymbol{q})\mathbf{H}^{-1}(\boldsymbol{q})[\boldsymbol{\tau} - \boldsymbol{n}(\boldsymbol{q}, \dot{\boldsymbol{q}})] \,. \tag{8.39}$$

Define the *control input* function

$$\boldsymbol{u} = \ddot{\boldsymbol{r}} + \dot{\mathbf{T}}\dot{\boldsymbol{q}} + \mathbf{T}(\boldsymbol{q})\mathbf{H}^{-1}(\boldsymbol{q})[\boldsymbol{\tau} - \boldsymbol{n}(\boldsymbol{q}, \dot{\boldsymbol{q}})] \,. \tag{8.40}$$

Now we may define a state $\boldsymbol{y}(t) \in R^{2p}$ by $\boldsymbol{y} = (\boldsymbol{\xi}\, \dot{\boldsymbol{\xi}})$ and write the manipulator dynamics as

$$\dot{\boldsymbol{y}} = \begin{pmatrix} \mathbf{0} & \mathbf{I}_\mathrm{p} \\ \mathbf{0} & \mathbf{0} \end{pmatrix} \boldsymbol{y} + \begin{pmatrix} \mathbf{0} \\ \mathbf{I}_\mathrm{p} \end{pmatrix} \boldsymbol{u} \,. \tag{8.41}$$

This is a linear state-space system of the form

$$\dot{\boldsymbol{y}} = \mathbf{A}\boldsymbol{y} + \mathbf{B}\boldsymbol{u} \,, \tag{8.42}$$

driven by the control input $\boldsymbol{u}$. Due to the special form of $\mathbf{A}$ and $\mathbf{B}$, this system is called the *Brunovsky canonical form* and it is always controllable from $\boldsymbol{u}(t)$.

Since (8.40) is said to be a *linearizing transformation* for the manipulator dynamic equation, one may invert this transformation to obtain the joint torque

$$\boldsymbol{\tau} = \mathbf{H}(\boldsymbol{q})\mathbf{T}^{+}(\boldsymbol{q})(\boldsymbol{u} - \ddot{\boldsymbol{r}} - \dot{\mathbf{T}}\dot{\boldsymbol{q}}) + \boldsymbol{n}(\boldsymbol{q}, \dot{\boldsymbol{q}}) \,, \tag{8.43}$$

where $\mathbf{T}^{+}$ denotes the *Moore–Penrose inverse* of the transformation matrix $\mathbf{T}(\boldsymbol{q})$.

In the special case $\boldsymbol{\xi} = \boldsymbol{e}_\mathrm{q}(t)$, and if we select $\boldsymbol{u}(t)$ so that (8.41) is stable by the PD feedback $\boldsymbol{u} = -\mathbf{K}_\mathrm{P}\boldsymbol{\xi} - \mathbf{K}_\mathrm{V}\dot{\boldsymbol{\xi}}$, then $\mathbf{T} = -\mathbf{I}_n$ and the control input torque $\boldsymbol{\tau}(t)$ defined by (8.43) makes the manipulator move in such a way that $\boldsymbol{y}(t)$ goes to zero. In this case, the feedback linearizing control and the inverse dynamics control become the same.

8.5.3 Passivity-Based Control

This method explicitly uses the passivity properties of the Lagrangian system [8.31, 32]. In comparison with the inverse dynamics method, passivity-based controllers are expected to have better robust properties because they do not rely on the exact cancellation of the manipulator nonlinearities. The passivity-based control input is given by

$$\begin{aligned} \dot{\boldsymbol{q}}_\mathrm{r} &= \dot{\boldsymbol{q}}_\mathrm{d} + \alpha \boldsymbol{e}_\mathrm{q} \,, \quad \alpha > 0 \,, \\ \boldsymbol{\tau} &= \mathbf{H}(\boldsymbol{q})\ddot{\boldsymbol{q}}_\mathrm{r} + \mathbf{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}}_\mathrm{r} + \boldsymbol{\tau}_\mathrm{g}(\boldsymbol{q}) + \mathbf{K}_\mathrm{V}\dot{\boldsymbol{e}}_\mathrm{q} + \mathbf{K}_\mathrm{P}\boldsymbol{e}_\mathrm{q} \,. \end{aligned} \tag{8.44}$$

With (8.44), we obtain the following closed-loop system

$$\mathbf{H}(\boldsymbol{q})\dot{\boldsymbol{s}}_\mathrm{q} + \mathbf{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\boldsymbol{s}_\mathrm{q} + \mathbf{K}_\mathrm{V}\dot{\boldsymbol{e}}_\mathrm{q} + \mathbf{K}_\mathrm{P}\boldsymbol{e}_\mathrm{q} = \mathbf{0} \,, \tag{8.45}$$

where $\boldsymbol{s}_\mathrm{q} = \dot{\boldsymbol{e}}_\mathrm{q} + \alpha \boldsymbol{e}_\mathrm{q}$. Let us choose a Lyapunov function $V(\boldsymbol{y}, t)$ as follows

$$V = \frac{1}{2}\boldsymbol{y}^\mathrm{T} \begin{pmatrix} \alpha\mathbf{K}_\mathrm{V} + \mathbf{K}_\mathrm{P} + \alpha^2\mathbf{H} & \alpha\mathbf{H} \\ \alpha\mathbf{H} & \mathbf{H} \end{pmatrix} \boldsymbol{y} = \frac{1}{2}\boldsymbol{y}^\mathrm{T}\mathbf{P}\boldsymbol{y} \,. \tag{8.46}$$

Avidbots Ex1016
Page 218 of 2154

Since the above equation is positive definite, it has a unique equilibrium at the origin, i. e., $\boldsymbol{y} = (\boldsymbol{e}_{\mathrm{q}}^{\mathrm{T}}, \dot{\boldsymbol{e}}_{\mathrm{q}}^{\mathrm{T}})^{\mathrm{T}} = \boldsymbol{0}$. Moreover, V can be bounded by

$$\sigma_{\mathrm{m}}\|\boldsymbol{y}\|^2 \leq \boldsymbol{y}^{\mathrm{T}}\mathbf{P}\boldsymbol{y} \leq \sigma_{\mathrm{M}}\|\boldsymbol{y}\|^2 , \quad \sigma_{\mathrm{M}} \geq \sigma_{\mathrm{m}} > 0 . \tag{8.47}$$

The time derivative of V gives

$$\dot{V} = -\dot{\boldsymbol{e}}_{\mathrm{q}}^{\mathrm{T}}\mathbf{K}_{\mathrm{V}}\dot{\boldsymbol{e}}_{\mathrm{q}} - \alpha \boldsymbol{e}_{\mathrm{q}}\mathbf{K}_{\mathrm{P}}\boldsymbol{e}_{\mathrm{q}} = -\boldsymbol{y}^{\mathrm{T}}\mathbf{Q}\boldsymbol{y} < 0 , \tag{8.48}$$

where $\mathbf{Q} = \mathrm{diag}[\alpha\mathbf{K}_{\mathrm{P}}, \mathbf{K}_{\mathrm{V}}]$. Since $\mathbf{Q}$ is positive definite and quadratic in $\boldsymbol{y}$, it can be also bounded by

$$\kappa_{\mathrm{m}}\|\boldsymbol{y}\|^2 \leq \boldsymbol{y}^{\mathrm{T}}\mathbf{Q}\boldsymbol{y} \leq \kappa_{\mathrm{M}}\|\boldsymbol{y}\|^2 , \quad \kappa_{\mathrm{M}} \geq \kappa_{\mathrm{m}} > 0 . \tag{8.49}$$

Then, from the bound of the Lyapunov function V, we get

$$\dot{V} \leq -\kappa_{\mathrm{m}}\|\boldsymbol{y}\|^2 = -2\eta V , \quad \eta = \frac{\kappa_{\mathrm{m}}}{\sigma_{\mathrm{M}}} , \tag{8.50}$$

which finally yields

$$V(t) \leq V(0)e^{-2\eta t} . \tag{8.51}$$

It has been shown that the value of α affects the tracking result dramatically [8.33]. The manipulator tends to vibrate for small values of α. Larger values of α allow better tracking performance and protect $\boldsymbol{s}_{\mathrm{q}}$ from being spoiled by the velocity measurement noise when the position error is small. In [8.34], it was suggested that

$$\mathbf{K}_{\mathrm{P}} = \alpha\mathbf{K}_{\mathrm{V}} \tag{8.52}$$

be used for quadratic optimization.

8.5.4 Summary

In this section, we have reviewed some of the model-based motion control methods proposed to date. Under some control approaches, the closed-loop system has either asymptotic stability or globally exponential stability. However, such ideal performance cannot be obtained in practical implementation because factors such as sampling rate, measurement noise, disturbances, and unmodeled dynamics will limit the achievable gain and the performance of the control algorithms [8.33, 35, 36].

8.6 Computed-Torque Control

Through the years many kinds of robot control schemes have been proposed. Most of these can be considered as special cases of the class of computed-torque control (Fig. 8.5) which is the technique of applying feedback linearization to nonlinear systems in general [8.37, 38]. In the section, computed-torque control will be first introduced, and its variant, so-called computed-torque-like control, will be introduced later.

8.6.1 Computed-Torque Control

Consider the control input (8.29)

$$\boldsymbol{\tau} = \mathbf{H}(\boldsymbol{q})\boldsymbol{v} + \mathbf{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{\tau}_{\mathrm{g}}(\boldsymbol{q}) ,$$

which is also known as computed-torque control; it consists of an inner nonlinear compensation loop and an outer loop with an exogenous control signal v. Substituting this control law into the dynamical model of the robot manipulator, it follows that

$$\ddot{\boldsymbol{q}} = \boldsymbol{v} . \tag{8.53}$$

It is important to note that this control input converts a complicated nonlinear controller design problem into a simple design problem for a linear system consisting of n decoupled subsystems. One approach to the outer-loop control $\boldsymbol{v}$ is propositional–derivative (PD) feedback, as in (8.30)

$$\boldsymbol{v} = \ddot{\boldsymbol{q}}_{\mathrm{d}} + \mathbf{K}_{\mathrm{V}}\dot{\boldsymbol{e}}_{\mathrm{q}} + \mathbf{K}_{\mathrm{P}}\boldsymbol{e}_{\mathrm{q}} ,$$

in which case the overall control input becomes

$$\boldsymbol{\tau} = \mathbf{H}(\boldsymbol{q})(\ddot{\boldsymbol{q}}_{\mathrm{d}} + \mathbf{K}_{\mathrm{V}}\dot{\boldsymbol{e}}_{\mathrm{q}} + \mathbf{K}_{\mathrm{P}}\boldsymbol{e}_{\mathrm{q}}) + \mathbf{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{\tau}_{\mathrm{g}}(\boldsymbol{q}) ,$$

and the resulting linear error dynamics are

$$\ddot{\boldsymbol{e}}_{\mathrm{q}} + \mathbf{K}_{\mathrm{V}}\dot{\boldsymbol{e}}_{\mathrm{q}} + \mathbf{K}_{\mathrm{P}}\boldsymbol{e}_{\mathrm{q}} = \boldsymbol{0} . \tag{8.54}$$

According to linear system theory, convergence of the tracking error to zero is guaranteed [8.29, 39].

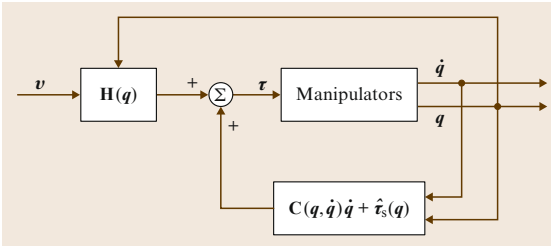


Fig. 8.5 Computed-torque control

Remark 8.2

One usually lets $\mathbf{K}_{\mathrm{V}}$ and $\mathbf{K}_{\mathrm{P}}$ be $(n \times n)$ diagonal positive-definite gain matrices (i. e., $\mathbf{K}_{\mathrm{V}} = \mathrm{diag}(k_{\mathrm{V},1},$

Avidbots Ex1016
Page 219 of 2154

$\cdots, k_{V,n}) > \mathbf{0}$, $\mathbf{K}_P = \mathrm{diag}(k_{P,1}, \cdots, k_{P,n}) > \mathbf{0}$) to guarantee the stability of the error system. However, the format of the foregoing control never leads to *independent joint control* because the outer-loop multiplier $\mathbf{H}(\boldsymbol{q})$ and the full nonlinear compensation term $\mathbf{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{\tau}_g(\boldsymbol{q})$ in the inner loop scramble all joint signals among different control channels.

8.6.2 Computed-Torque-Like Control

It is worth noting that the implementation of computed-torque control requires that parameters of the dynamical model are accurately known and the control input is computed in real time. In order to avoid those problems, several variations of this control scheme have been proposed, for example, computed-torque-like control. An entire class of computed-torque-like controllers can be obtained by modifying the computed-torque control as

$$\boldsymbol{\tau} = \hat{\mathbf{H}}(\boldsymbol{q})\boldsymbol{v} + \hat{\mathbf{C}}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \hat{\boldsymbol{\tau}}(\boldsymbol{q}) \,, \tag{8.55}$$

where $\hat{}$ represents the computed or nominal value and indicates that the theoretically exact feedback linearization cannot be achieved in practice due to the uncertainty in the systems. The overall control scheme is depicted in Fig. 8.6.

Computed-Torque-Like Control with Variable-Structure Compensation

Since there is parametric uncertainty, compensation is required in the outer-loop design to achieve trajectory tracking. The following shows a computed-torque-like control with variable-structure compensation

$$\boldsymbol{v} = \ddot{\boldsymbol{q}}_d + \mathbf{K}_V \dot{\boldsymbol{e}}_q + \mathbf{K}_P \boldsymbol{e}_q + \Delta \boldsymbol{v} \,, \tag{8.56}$$

where the variable-structure compensation $\Delta \boldsymbol{v}$ is devised as

$$\Delta \boldsymbol{v} = \begin{cases} -\rho(\boldsymbol{x}, t)\dfrac{\mathbf{B}^T\mathbf{P}\boldsymbol{x}}{\| \mathbf{B}^T\mathbf{P}\boldsymbol{x} \|} & \text{if} \quad \| \mathbf{B}^T\mathbf{P}\boldsymbol{x} \| \neq 0 \,, \\ 0 & \text{if} \quad \| \mathbf{B}^T\mathbf{P}\boldsymbol{x} \| = 0 \,, \end{cases} \tag{8.57}$$

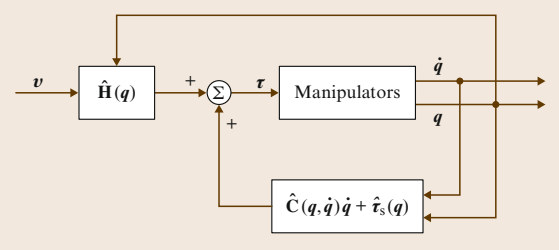


Fig. 8.6 Computed-torque-like control

where $\boldsymbol{x} = (\boldsymbol{e}_q^T, \dot{\boldsymbol{e}}_q^T)^T$, $\mathbf{B} = (\mathbf{0}, \mathbf{I}_n)^T$, $\mathbf{P}$ is a $(2n \times 2n)$ symmetric positive-definite matrix satisfying

$$\mathbf{P}\mathbf{A} + \mathbf{A}^T\mathbf{P} = -\mathbf{Q} \,, \tag{8.58}$$

with $\mathbf{A}$ being defined as

$$\mathbf{A} = \begin{pmatrix} \mathbf{0} & \mathbf{I}_n \\ -\mathbf{K}_P & -\mathbf{K}_V \end{pmatrix} \,, \tag{8.59}$$

$\mathbf{Q}$ being any appropriate $(2n \times 2n)$ symmetric positive-definite matrix,

$$\rho(\boldsymbol{x}, t) = \frac{1}{1-\alpha}[\alpha\beta + \| \mathbf{K} \| \| \boldsymbol{x} \| + \bar{\mathbf{H}}\phi(\boldsymbol{x}, t)] \,, \tag{8.60}$$

where α and β are positive constants such that $\| \mathbf{H}^{-1}(\boldsymbol{q})\hat{\mathbf{H}}(\boldsymbol{q}) - \mathbf{I}_n \| \leq \alpha < 1$ for all $\boldsymbol{q} \in R^n$ and $\sup_{t \in [0,\infty)} \| \ddot{\boldsymbol{q}}_d(t) \| < \beta$, respectively, $\mathbf{K}$ is the $(n \times 2n)$ matrix defined as $\mathbf{K} = [\mathbf{K}_P \mathbf{K}_V]$, $\bar{\lambda}_H$ being a positive constant such that $\| \mathbf{H}^{-1}(\boldsymbol{q}) \| \leq \bar{\lambda}_H$ for all $\boldsymbol{q} \in R^n$, and the function ϕ being defined as

$$\begin{aligned} &\| [\hat{\mathbf{C}}(\boldsymbol{q}, \dot{\boldsymbol{q}}) - \mathbf{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})]\dot{\boldsymbol{q}} + [\hat{\boldsymbol{\tau}}_g(\boldsymbol{q}) - \boldsymbol{\tau}_g(\boldsymbol{q})] \| \\ &\quad \leq \phi(\boldsymbol{x}, t) \,. \end{aligned} \tag{8.61}$$

Convergence of the tracking error to zero can be shown using the Lyapunov function

$$V = \boldsymbol{x}^T\mathbf{P}\boldsymbol{x} \,, \tag{8.62}$$

following the stability analysis in [8.5, 40].

Remarks 8.3

- By Property 8.1 in Sect. 8.1, there exist positive constants $\bar{\lambda}_H$ and $\bar{\lambda}_h$ such that $\bar{\lambda}_h \leq \| \mathbf{H}^{-1}(\boldsymbol{q}) \| \leq \bar{\lambda}_H$ for all $\boldsymbol{q} \in R^n$. If we choose

$$\hat{\mathbf{H}} = \frac{1}{c}\mathbf{I}_n \,, \quad \text{where} \quad c = \frac{\bar{\lambda}_H + \bar{\lambda}_h}{2} \,, \tag{8.63}$$

it can be shown that

$$\| \mathbf{H}^{-1}(\boldsymbol{q})\hat{\mathbf{H}}(\boldsymbol{q}) - \mathbf{I}_n \| \leq \frac{\bar{\lambda}_H - \bar{\lambda}_h}{\bar{\lambda}_H + \bar{\lambda}_h} \equiv \alpha < 1 \,, \tag{8.64}$$

which indicates that there is always at least one choice of $\hat{\mathbf{H}}$ for some $\alpha < 1$.

- Due to the discontinuity in $\Delta \boldsymbol{v}$, chattering phenomenon may occur when the control scheme is applied. It is worth noting that chattering is often undesirable since the high-frequency component in

Avidbots Ex1016
Page 220 of 2154

the control can excite unmodeled dynamic effect (such as joint flexibility) [8.6, 29, 38]. In order to avoid chattering, the variable-structure compensation can be modified to become smooth, i. e.,

$$\Delta \boldsymbol{v} = \begin{cases} -\rho(\boldsymbol{x},t)\dfrac{\mathbf{B}^{\mathrm{T}}\mathbf{P}\boldsymbol{x}}{\| \mathbf{B}^{\mathrm{T}}\mathbf{P}\boldsymbol{x} \|} & \text{if } \| \mathbf{B}^{\mathrm{T}}\mathbf{P}\boldsymbol{x} \| > \varepsilon \,, \\ -\dfrac{\rho(\boldsymbol{x},t)}{\varepsilon}\mathbf{B}^{\mathrm{T}}\mathbf{P}\boldsymbol{x} & \text{if } \| \mathbf{B}^{\mathrm{T}}\mathbf{P}\boldsymbol{x} \| \leq \varepsilon \,, \end{cases} \tag{8.65}$$

where ε is a positive constant and is used as the boundary layer. Following this modification, convergence of tracking errors to a residual set can be ensured, and the size of this residual set can be made smaller by use of a smaller value of ε.

Computed-Torque-Like Control with Independent-Joint Compensation

Obviously, the previous compensation scheme is centralized, which implies that the online computation load is heavy and high-cost hardware is required for practical implementation. In order to solve this problem, a scheme with independent-joint compensation is introduced below. In this scheme, a computed-torque-like control is designed with estimates as

$$\hat{\mathbf{H}}(\boldsymbol{q}) = \mathbf{I} \,, \quad \hat{\mathbf{C}}(\boldsymbol{q},\dot{\boldsymbol{q}}) = 0 \,, \quad \hat{\boldsymbol{\tau}}(\boldsymbol{q}) = \mathbf{0} \,. \tag{8.66}$$

Then, we use the outer-loop $\boldsymbol{v}$ as

$$\boldsymbol{v} = \mathbf{K}_{\mathrm{V}}\dot{\boldsymbol{e}}_{\mathrm{q}} + \mathbf{K}_{\mathrm{P}}\boldsymbol{e}_{\mathrm{q}} + \Delta\boldsymbol{v} \,, \tag{8.67}$$

where the positive constants $\mathbf{K}_{\mathrm{P}}$ and $\mathbf{K}_{\mathrm{V}}$ are selected to be sufficiently large, and the i-th component $\Delta\boldsymbol{v}_i$ of $\Delta\boldsymbol{v} = (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n)^{\mathrm{T}}$ is

$$\Delta\boldsymbol{v}_i = \begin{cases} -\left[\boldsymbol{\beta}^{\mathrm{T}}\boldsymbol{w}(\boldsymbol{q}_{\mathrm{d}},\dot{\boldsymbol{q}}_{\mathrm{d}})\right]^2 \dfrac{s_i}{\varepsilon_i} \\ \quad \text{if} \quad |\, s_i \,| \leq \dfrac{\varepsilon_i}{\boldsymbol{\beta}^{\mathrm{T}}\boldsymbol{w}(\boldsymbol{q}_{\mathrm{d}},\dot{\boldsymbol{q}}_{\mathrm{d}})} \,, \\ -\boldsymbol{\beta}^{\mathrm{T}}\boldsymbol{w}(\boldsymbol{q}_{\mathrm{d}},\dot{\boldsymbol{q}}_{\mathrm{d}})\dfrac{s_i}{|s_i|} \\ \quad \text{if} \quad |\, s_i \,| > \dfrac{\varepsilon_i}{\boldsymbol{\beta}^{\mathrm{T}}\boldsymbol{w}(\boldsymbol{q}_{\mathrm{d}},\dot{\boldsymbol{q}}_{\mathrm{d}})} \,. \end{cases} \tag{8.68}$$

In this compensation, $s_i = \dot{e}_{\mathrm{q},i} + \lambda_i e_{\mathrm{q},i}$, $i \in \{1, \ldots, n\}$, and $\lambda_i, i \in \{1, \ldots, n\}$ are positive constants. Furthermore, following the properties of robot manipulators, we have

$$\begin{aligned} &\| \mathbf{H}(\boldsymbol{q})\ddot{\boldsymbol{q}}_{\mathrm{d}} + \mathbf{C}(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}}_{\mathrm{d}} + \boldsymbol{\tau}_{\mathrm{g}}(\boldsymbol{q}) \| \\ &\quad \leq \beta_1 + \beta_2 \| \boldsymbol{q} \| + \| \dot{\boldsymbol{q}} \| = \boldsymbol{\beta}^{\mathrm{T}}\boldsymbol{w}(\boldsymbol{q},\dot{\boldsymbol{q}}) \end{aligned}$$

for some suitable positive constants β_1, β_2, and β_3, where $\boldsymbol{\beta} = (\beta_1, \beta_2, \beta_3)^{\mathrm{T}}$ and

$$\boldsymbol{w}(\boldsymbol{q},\dot{\boldsymbol{q}}) = [1, \| \boldsymbol{q} \|, \| \dot{\boldsymbol{q}} \|]^{\mathrm{T}} \,. \tag{8.69}$$

Finally, ε_i, $i \in \{1, \ldots, n\}$, is the variable length of the boundary layer, satisfying

$$\dot{\varepsilon}_i = -g_i\,\varepsilon_i \,, \quad \varepsilon(0) > 0 \,, \quad g_i > 0 \,. \tag{8.70}$$

It is worth pointing out that the term $\boldsymbol{w}$ in this control scheme is devised as the desired compensation rather than feedback. Furthermore, this scheme is in a form of independent-joint control and hence has the advantages introduced before. The convergence of the tracking error to zero can be shown using the Lyapunov function

$$V = \frac{1}{2}\left(\boldsymbol{e}_{\mathrm{q}}^{\mathrm{T}}\dot{\boldsymbol{e}}_{\mathrm{q}}^{\mathrm{T}}\right)\begin{pmatrix}\lambda\mathbf{K}_{\mathrm{P}} & \mathbf{H} \\ \mathbf{H} & \lambda\mathbf{H}\end{pmatrix}\begin{pmatrix}\boldsymbol{e}_{\mathrm{q}} \\ \dot{\boldsymbol{e}}_{\mathrm{q}}\end{pmatrix} + \sum_{i=1}^{n} g_i^{-1}\varepsilon_i \,, \tag{8.71}$$

whose time derivative along the trajectory of the closed-loop systems follows

$$\dot{V} = -\alpha \left| \begin{pmatrix}\boldsymbol{e}_{\mathrm{q}} \\ \dot{\boldsymbol{e}}_{\mathrm{q}}\end{pmatrix} \right|^2 \,, \tag{8.72}$$

with α being some positive constant, if sufficiently large $\mathbf{K}_{\mathrm{P}}$ and γ are used. The detailed analysis of stability, which requires Properties 8.3 and 8.4, can be found in [8.13].

Remark 8.3

Similar to computed-torque-like control with variable-structure compensation, we can consider the nonzero boundary layer as

$$\dot{\varepsilon}_i = -g_i\varepsilon_i \,, \quad \varepsilon(0) > 0 \,, \quad g_i \,,\, \alpha_i > 0 \,. \tag{8.73}$$

Following this modification, tracking errors converge to a residual set. The size of this residual set can be made smaller by the use of a smaller value of ε (i. e., smaller α_i).

For the task of point-to-point control, one PD controller with gravity compensation is designed with the estimates

$$\hat{\mathbf{H}}(\boldsymbol{q}) = \mathbf{I} \,, \quad \hat{\mathbf{C}}(\boldsymbol{q},\dot{\boldsymbol{q}}) = \mathbf{0} \,, \quad \hat{\boldsymbol{\tau}}_{\mathrm{g}}(\boldsymbol{q}) = \boldsymbol{\tau}_{\mathrm{g}}(\boldsymbol{q}) \,, \tag{8.74}$$

with $\boldsymbol{\tau}_{\mathrm{g}}(\boldsymbol{q})$ being the *gravity term* of the dynamical model of the robot manipulators. Then, we use the outer-loop $\boldsymbol{v}$ as

$$\boldsymbol{v} = \mathbf{K}_{\mathrm{V}}\dot{\boldsymbol{e}}_{\mathrm{q}} + \mathbf{K}_{\mathrm{P}}\boldsymbol{e}_{\mathrm{q}} \,, \tag{8.75}$$

such that the control input becomes

$$\boldsymbol{\tau} = \mathbf{K}_{\mathrm{V}}\dot{\boldsymbol{e}}_{\mathrm{q}} + \mathbf{K}_{\mathrm{P}}\boldsymbol{e}_{\mathrm{q}} + \boldsymbol{\tau}_{\mathrm{g}}(\boldsymbol{q}) \,. \tag{8.76}$$

Avidbots Ex1016
Page 221 of 2154

This scheme is much simpler to implement than the exact computed-torque controller. The convergence of the tracking error to zero can be shown using the Lyapunov function

$$V = \frac{1}{2}\dot{\boldsymbol{e}}_{\mathrm{q}}^{\mathrm{T}}\mathbf{H}(\boldsymbol{q})\dot{\boldsymbol{e}}_{\mathrm{q}} + \frac{1}{2}\boldsymbol{e}_{\mathrm{q}}^{\mathrm{T}}\mathbf{K}_{\mathrm{P}}\boldsymbol{e}_{\mathrm{q}}\,, \tag{8.77}$$

whose time derivative along the solution trajectories of the closed-loop system is

$$\dot{V} = -\dot{\boldsymbol{e}}_{\mathrm{q}}^{\mathrm{T}}\mathbf{K}_{\mathrm{V}}\dot{\boldsymbol{e}}_{\mathrm{q}}\,. \tag{8.78}$$

The detailed analysis of stability is given in [8.12]. It is necessary to note that this result is for the case of regulation, rather than for the case of tracking, since the former theoretical base, which relies on LaSalle's lemma, requires the system be autonomous (time invariant) [8.38, 41, 42].

Remark 8.4

If we neglect gravity in the dynamical model of the robot manipulators, then the gravity estimation can be omitted here, i. e., $\hat{\boldsymbol{\tau}}_{\mathrm{g}}(\boldsymbol{q}) = \mathbf{0}$, so that the control law becomes

$$\boldsymbol{\tau} = \boldsymbol{v} = \mathbf{K}_{\mathrm{V}}\dot{\boldsymbol{e}}_{\mathrm{q}} + \mathbf{K}_{\mathrm{P}}\boldsymbol{e}_{\mathrm{q}}\,, \tag{8.79}$$

which leads to pure PD control. The gain matrices, $\mathbf{K}_{\mathrm{P}}$ and $\mathbf{K}_{\mathrm{V}}$, can be selected to be diagonal so that this PD control is in a form of independent-joint control developed based on the multijoint dynamic model.

8.6.3 Summary

In this section, we have presented two control schemes: the computed-torque and computed-torque-like control. The former transforms a multi-input multi-output (MIMO) nonlinear robotic system into a very simple decoupled linear closed-loop system whose control design is a well-established problem. Since the practical implementation of the former control requires preknowledge of all the manipulator parameters and its payload, which may not be realistic, the latter was introduced to relax the foregoing requirement and still to achieve the objective of tracking subject to system's uncertainty.

Further Reading

The PD control with different feedforward compensation for tracking control is investigated in [8.43]. An adaptive control scheme based on PD control is presented in [8.19].

8.7 Adaptive Control

An adaptive controller differs from an ordinary controller in that the controller parameters are time varying, and there is a mechanism for adjusting these parameters online based on some signals in the closed-loop systems. By the use of such control scheme, the control goal can be achieved even if there is parametric uncertainty in the plant. In this section, we will introduce several adaptive control schemes to deal with the case of imperfect knowledge of dynamical parameters of the robot manipulators. The control performance of those adaptive control schemes, including adaptive computed-torque control, adaptive inertia-related control, adaptive control based on passivity, and adaptive control with desired compensation, are basically derived from Property 8.5. Finally, the condition of persistent excitation, which is important in parameter convergence, will be addressed.

8.7.1 Adaptive Computed-Torque Control

The computed-torque control scheme is appealing, since it allows the designer to transform a MIMO highly coupled nonlinear system into a very simple decoupled linear system, whose control design is a well-established problem. However, this method of feedback linearization relies on perfect knowledge of the system parameters, and failure to have this will cause erroneous parametric estimates, resulting in a mismatch term in the closed-loop model of the error system. That term can be interpreted as a nonlinear perturbation acting at the input of the closed-loop system. In order to solve the problem due to parametric uncertainty, we instead consider the inverse dynamics method with parameter estimates of

$$\boldsymbol{\tau} = \hat{\mathbf{H}}(\boldsymbol{q})(\ddot{\boldsymbol{q}}_{\mathrm{d}} + \mathbf{K}_{\mathrm{V}}\dot{\boldsymbol{e}}_{\mathrm{q}} + \mathbf{K}_{\mathrm{P}}\boldsymbol{e}_{\mathrm{q}}) + \hat{\mathbf{C}}(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \hat{\boldsymbol{\tau}}_{\mathrm{g}}(\boldsymbol{q})\,, \tag{8.80}$$

where $\hat{\mathbf{H}}$, $\hat{\mathbf{C}}$, $\hat{\boldsymbol{\tau}}_{\mathrm{g}}$ have the same functional form as $\mathbf{H}$, $\mathbf{C}$, $\boldsymbol{\tau}_{\mathrm{g}}$. From Property 8.5 of the dynamics model, we have

$$\hat{\mathbf{H}}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \hat{\mathbf{C}}(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \hat{\boldsymbol{\tau}}_{\mathrm{g}}(\boldsymbol{q}) = \mathbf{Y}(\boldsymbol{q},\dot{\boldsymbol{q}},\ddot{\boldsymbol{q}})\hat{\boldsymbol{a}}\,, \tag{8.81}$$

where $\mathbf{Y}(\boldsymbol{q},\dot{\boldsymbol{q}},\ddot{\boldsymbol{q}})$, called the regressor, is a known $(n\times r)$ function matrix and $\boldsymbol{a}$ is the $(r\times 1)$ vector that summarizes all the estimated parameters. Substituting this

control input $\boldsymbol{\tau}$ into the manipulator dynamics gives the following closed-loop error model

$$\hat{\mathbf{H}}(\boldsymbol{q})(\ddot{\boldsymbol{e}}_{\mathrm{q}} + \mathbf{K}_{\mathrm{V}}\dot{\boldsymbol{e}}_{\mathrm{q}} + \mathbf{K}_{\mathrm{P}}\boldsymbol{e}_{\mathrm{q}}) = \mathbf{Y}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}})\tilde{\boldsymbol{a}} \,, \tag{8.82}$$

where $\tilde{\boldsymbol{a}} = \hat{\boldsymbol{a}} - \boldsymbol{a}$. In order to acquire an appropriate adaptive law, we first assume that the acceleration term $\ddot{\boldsymbol{q}}$ is measurable, and that the estimated inertia matrix $\hat{\mathbf{H}}(\boldsymbol{q})$ is never singular. Now, for convenience, the error equation is rewritten as

$$\dot{\boldsymbol{x}} = \mathbf{A}\boldsymbol{x} + \mathbf{B}\hat{\mathbf{H}}^{-1}(\boldsymbol{q})\mathbf{Y}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}})\tilde{\boldsymbol{a}} \,, \tag{8.83}$$

with $\boldsymbol{x} = \left(\boldsymbol{e}_{\mathrm{q}}^{\mathrm{T}}, \dot{\boldsymbol{e}}_{\mathrm{q}}^{\mathrm{T}}\right)^{\mathrm{T}}$,

$$\mathbf{A} = \begin{pmatrix} \mathbf{0}_n & \mathbf{I}_n \\ -\mathbf{K}_{\mathrm{P}} & -\mathbf{K}_{\mathrm{V}} \end{pmatrix} \,, \quad \mathbf{B} = \begin{pmatrix} \mathbf{0}_n \\ \mathbf{I}_n \end{pmatrix} \,. \tag{8.84}$$

The adaptive law is considered as

$$\dot{\hat{\boldsymbol{a}}} = -\boldsymbol{\Gamma}^{-1}\mathbf{Y}^{\mathrm{T}}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}})\hat{\mathbf{H}}^{-1}(\boldsymbol{q})\mathbf{B}^{\mathrm{T}}\mathbf{P}\boldsymbol{x} \,, \tag{8.85}$$

where $\boldsymbol{\Gamma}$ is an $(r \times r)$ positive-definite constant matrix, and $\mathbf{P}$ is a $(2n \times 2n)$ symmetric positive-definite constant matrix satisfying

$$\mathbf{P}\mathbf{A} + \mathbf{A}^{\mathrm{T}}\mathbf{P} = -\mathbf{Q} \,, \tag{8.86}$$

with $\mathbf{Q}$ being a symmetric positive-definite constant matrix with coherent dimension. In this adaptive law, we made two assumptions:

- The joint acceleration $\ddot{\boldsymbol{q}}$ is measurable, and
- The bounded range of the unknown parameter is available.

The first assumption is to ensure that the regressor $\mathbf{Y}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}})$ is known a priori, whereas the second assumption is to allow one to keep the estimate $\hat{\mathbf{H}}(\boldsymbol{q})$ nonsingular by restricting the estimated parameter $\hat{\boldsymbol{a}}$ to lie within a range about the true parameter value.

Convergence of the tracking error and maintaining boundedness of all internal signals can actually be guaranteed by Lyapunov stability theory with the Lyapunov function

$$\dot{V} = -\boldsymbol{x}^{\mathrm{T}}\mathbf{Q}\boldsymbol{x} \,. \tag{8.87}$$

Detailed stability analysis is given in [8.2].

Remark 8.5
For practical and theoretical reasons, the first assumption above is hardly acceptable. In most cases, it is not easy to obtain an accurate measure of acceleration; the robustness of the above adaptive control scheme with respect to such a disturbance has to be established. Moreover, from a pure theoretical viewpoint, measuring $\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}$ means that not only do we need the whole system state vector, but we also need its derivative.

8.7.2 Adaptive Inertia-Related Control

Another adaptive control scheme is now introduced. This proposed scheme does not require the measurement of the manipulator's acceleration nor does it require inversion of the estimated inertia matrix. Hence, the drawbacks of the adaptive computed-torque control scheme are avoided. Let us consider the control input

$$\boldsymbol{\tau} = \hat{\mathbf{H}}(\boldsymbol{q})\dot{\boldsymbol{v}} + \hat{\mathbf{C}}(\boldsymbol{q}, \dot{\boldsymbol{q}})\boldsymbol{v} + \hat{\boldsymbol{\tau}}_{\mathrm{g}}(\boldsymbol{q}) + \mathbf{K}_{\mathrm{D}}\boldsymbol{s} \,, \tag{8.88}$$

where the auxiliary signals $\boldsymbol{v}$ and $\boldsymbol{s}$ are defined as $\boldsymbol{v} = \dot{\boldsymbol{q}}_{\mathrm{d}} + \boldsymbol{\Lambda}\boldsymbol{e}_{\mathrm{q}}$ and $\boldsymbol{s} = \boldsymbol{v} - \dot{\boldsymbol{q}} = \dot{\boldsymbol{e}}_{\mathrm{q}} + \boldsymbol{\Lambda}\boldsymbol{e}_{\mathrm{q}}$, with $\boldsymbol{\Lambda}$ being an $(n \times n)$ positive-definite matrix. Following Property 8.5 of the dynamic model, we have

$$\mathbf{H}(\boldsymbol{q})\dot{\boldsymbol{v}} + \mathbf{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\boldsymbol{v} + \boldsymbol{\tau}_{\mathrm{g}}(\boldsymbol{q}) = \bar{\mathbf{Y}}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{v}, \dot{\boldsymbol{v}})\boldsymbol{a} \,, \tag{8.89}$$

where $\bar{\mathbf{Y}}(\cdot, \cdot, \cdot, \cdot)$ is an $(n \times r)$ matrix of known time functions. The formulation above is the same type of the parameter separation that was used in the formulation of the adaptive computed-torque control. Note that $\bar{\mathbf{Y}}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{v}, \dot{\boldsymbol{v}})$ is independent of the joint acceleration. Similar to the formulation above, we also have

$$\hat{\mathbf{H}}(\boldsymbol{q})\dot{\boldsymbol{v}} + \hat{\mathbf{C}}(\boldsymbol{q}, \dot{\boldsymbol{q}})\boldsymbol{v} + \hat{\boldsymbol{\tau}}_{\mathrm{g}}(\boldsymbol{q}) = \bar{\mathbf{Y}}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{v}, \dot{\boldsymbol{v}})\hat{\boldsymbol{a}} \,. \tag{8.90}$$

Substituting the control input into the equation of motion, it follows that

$$\begin{aligned} &\mathbf{H}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \mathbf{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{\tau}_{\mathrm{g}}(\boldsymbol{p}) \\ &\quad = \hat{\mathbf{H}}(\boldsymbol{q})\dot{\boldsymbol{v}} + \hat{\mathbf{C}}(\boldsymbol{q}, \dot{\boldsymbol{q}})\boldsymbol{v} + \hat{\boldsymbol{\tau}}_{\mathrm{g}}(\boldsymbol{q}) + \mathbf{K}_{\mathrm{D}}\boldsymbol{s} \,. \end{aligned}$$

Since $\ddot{\boldsymbol{q}} = \dot{\boldsymbol{v}} - \dot{\boldsymbol{s}}, \dot{\boldsymbol{q}} = \boldsymbol{v} - \boldsymbol{s}$, the previous result can be rewritten as

$$\mathbf{H}(\boldsymbol{q})\dot{\boldsymbol{s}} + \mathbf{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\boldsymbol{s} + \mathbf{K}_{\mathrm{D}}\boldsymbol{s} = \bar{\mathbf{Y}}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{v}, \dot{\boldsymbol{v}})\tilde{\boldsymbol{a}} \,, \tag{8.91}$$

where $\tilde{\boldsymbol{a}} = \boldsymbol{a} - \hat{\boldsymbol{a}}$. The adaptive law is considered as

$$\dot{\hat{\boldsymbol{a}}} = \boldsymbol{\Gamma}\bar{\mathbf{Y}}^{\mathrm{T}}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{v}, \dot{\boldsymbol{v}})\boldsymbol{s} \,. \tag{8.92}$$

The convergence of the tracking error to zero with boundedness on all internal signals can be shown

Avidbots Ex1016
Page 223 of 2154

through Lyapunov stability theory using the following Lyapunov-like function

$$V = \frac{1}{2}\boldsymbol{s}^{\mathrm{T}}\mathbf{H}(\boldsymbol{q})\boldsymbol{s} + \frac{1}{2}\tilde{\boldsymbol{a}}^{\mathrm{T}}\boldsymbol{\Gamma}^{-1}\tilde{\boldsymbol{a}}\,, \tag{8.93}$$

whose time derivative along the trajectories of the closed-loop systems can be found to be

$$\dot{V} = -\boldsymbol{s}^{\mathrm{T}}\mathbf{K}_{\mathrm{D}}\boldsymbol{s}\,. \tag{8.94}$$

The detailed stability analysis is given in [8.32].

Remark 8.6

- The restrictions for adaptive computed-torque control formerly seen have been removed here.
- The term $\mathbf{K}_{\mathrm{D}}\boldsymbol{s}$ introduces a PD-type linear stabilizing control action to the error system model.
- The estimated parameters converge to the true parameters provided the reference trajectory satisfies the condition of persistency of excitation,

$$\alpha_1\mathbf{I}_r \le \int_{t_0}^{t_0+t} \mathbf{Y}^{\mathrm{T}}(\boldsymbol{q}_{\mathrm{d}}, \dot{\boldsymbol{q}}_{\mathrm{d}}, \boldsymbol{v}, \dot{\boldsymbol{v}})\mathbf{Y}(\boldsymbol{q}_{\mathrm{d}}, \dot{\boldsymbol{q}}_{\mathrm{d}}, \boldsymbol{v}, \dot{\boldsymbol{v}})\,\mathrm{d}t \le \alpha_2\mathbf{I}_r\,,$$

for all t_0, where α_1, α_2, and t are all positive constants.

8.7.3 Adaptive Control Based on Passivity

Taking a physics viewpoint of control, we see that the concept of passivity has become popular for the development of adaptive control schemes. Here, we illustrate how the concept of passivity can be used to design a class of adaptive control laws for robot manipulators. First, we define an auxiliary filtered tracking error signal $\boldsymbol{r}$ as

$$\boldsymbol{r} = \mathbf{F}^{-1}(s)\boldsymbol{e}_{\mathrm{q}}\,, \tag{8.95}$$

where

$$\mathbf{F}^{-1}(s) = \left[s\mathbf{I}_n + \frac{1}{s}\mathbf{K}(s)\right]\,, \tag{8.96}$$

and s is the Laplace transform variable. The $(n\times n)$ matrix $\mathbf{K}(s)$ is chosen such that $\mathbf{F}(s)$ is a strictly proper, stable transfer function matrix. As in the preceding schemes, the adaptive control strategies has close ties to the ability to separate the known functions from the unknown constant parameters. We use the expression given above to define

$$\mathbf{Z}\boldsymbol{\varphi} = \mathbf{H}(\boldsymbol{q})[\ddot{\boldsymbol{q}}_{\mathrm{d}} + \mathbf{K}(s)\boldsymbol{e}_{\mathrm{q}}] + \boldsymbol{V}(\boldsymbol{q}, \dot{\boldsymbol{q}})\left[\dot{\boldsymbol{q}}_n + \frac{1}{s}\mathbf{K}(s)\boldsymbol{e}_{\mathrm{q}}\right] + \boldsymbol{\tau}_{\mathrm{g}}(\boldsymbol{q})\,,$$

where $\mathbf{Z}$ is a known $(n\times r)$ regression matrix and $\boldsymbol{\varphi}$ is a vector of unknown system parameters in the adaptive context. It is important to note that the above can be arranged such that $\mathbf{Z}$ and $\boldsymbol{r}$ do not depend on the measurement of the joint acceleration $\ddot{\boldsymbol{q}}$. The adaptive control scheme given here is called the passivity approach because the mapping of $-\boldsymbol{r} \to \mathbf{Z}\tilde{\boldsymbol{\varphi}}$ is constructed to be a passive mapping. That is, we develop an adaptive law such that

$$\int_0^t -\boldsymbol{r}^{\mathrm{T}}(\sigma)\mathbf{Z}(\sigma)\tilde{\boldsymbol{\varphi}}(\sigma)\mathrm{d}\sigma \ge -\beta \tag{8.97}$$

is satisfied for all time and for some positive scalar constant β. For this class of adaptive controllers, the control input is given by

$$\boldsymbol{\tau} = \mathbf{Z}\hat{\boldsymbol{\varphi}} + \mathbf{K}_{\mathrm{D}}\boldsymbol{r}\,, \tag{8.98}$$

Detailed analysis of stability is given in [8.44].

Remarks 8.4

- If $\mathbf{K}(s)$ is selected such that $\mathbf{H}(s)$ has a relative degree of one, $\mathbf{Z}$ and $\boldsymbol{r}$ will not depend on $\ddot{\boldsymbol{q}}$.
- Many types of control schemes can be generated from the adaptive passive control approach by selected different transfer function matrices $\mathbf{K}(s)$ in the definition of $\boldsymbol{r}$.
- Note that, by defining $\mathbf{K}(s) = s\boldsymbol{\Lambda}$ such that $\mathbf{F}(s) = (s\mathbf{I}_n + \boldsymbol{\Lambda})^{-1}$, we have the control input

$$\boldsymbol{\tau} = \mathbf{Z}\hat{\boldsymbol{\varphi}} - \mathbf{K}_{\mathrm{D}}\boldsymbol{r}\,,$$

with

$$\mathbf{Z}\hat{\boldsymbol{\varphi}} = \hat{\mathbf{H}}(\boldsymbol{q})(\ddot{\boldsymbol{q}}_{\mathrm{d}} + \boldsymbol{\Lambda}\dot{\boldsymbol{e}}_{\mathrm{q}}) + \hat{\mathbf{C}}(\boldsymbol{q}, \dot{\boldsymbol{q}})(\dot{\boldsymbol{q}}_{\mathrm{d}} + \boldsymbol{\Lambda}\boldsymbol{e}_{\mathrm{q}}) + \hat{\boldsymbol{\tau}}_{\mathrm{g}}(\boldsymbol{q})\,.$$

The adaptive law may be chosen as

$$\dot{\hat{\boldsymbol{\varphi}}} = \boldsymbol{\Gamma}\mathbf{Z}^{\mathrm{T}}(\dot{\boldsymbol{e}}_{\mathrm{q}} + \boldsymbol{\Lambda}\boldsymbol{e}_{\mathrm{q}})$$

to satisfy the condition of passive mapping. This indicates that adaptive inertia-related control can be viewed as a special case of adaptive passive control.

8.7.4 Adaptive Control with Desired Compensation

In order to implement the adaptive control scheme, one needs to calculate the elements of $\mathbf{Y}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}})$ in real time. However, this procedure may be excessively time consuming since it involves computations with highly nonlinear functions of joint position and velocity. Consequently, the real-time implementation of such a scheme is rather difficult. To overcome this difficulty, the adaptive control with desired compensation was proposed and is discussed here. In other words, the variables $\boldsymbol{q}$, $\dot{\boldsymbol{q}}$, and $\ddot{\boldsymbol{q}}$ are replaced with the desired ones, namely, $\boldsymbol{q}_\mathrm{d}$, $\dot{\boldsymbol{q}}_\mathrm{d}$, and $\ddot{\boldsymbol{q}}_\mathrm{d}$. Since the desired quantities are known in advance, all their corresponding calculations can be performed offline, which renders real-time implementation more plausible. Let us consider the control input

$$\boldsymbol{\tau} = \mathbf{Y}(\boldsymbol{q}_\mathrm{d}, \dot{\boldsymbol{q}}_\mathrm{d}, \ddot{\boldsymbol{q}}_\mathrm{d})\hat{\boldsymbol{a}} + k_\mathrm{a}\boldsymbol{s} + k_\mathrm{p}\boldsymbol{e}_\mathrm{q} + k_n\|\boldsymbol{e}_\mathrm{q}\|^2\boldsymbol{s}\,, \quad (8.99)$$

where the positive constants k_a, k_p, and k_n are sufficiently large, and the auxiliary signal $\boldsymbol{s}$ is defined as $\boldsymbol{s} = \dot{\boldsymbol{e}}_\mathrm{q} + \boldsymbol{e}_\mathrm{q}$. The adaptive law is considered as

$$\dot{\hat{\boldsymbol{a}}} = -\boldsymbol{\Gamma}\,\mathbf{Y}^\mathrm{T}(\boldsymbol{q}_\mathrm{d}, \dot{\boldsymbol{q}}_\mathrm{d}, \ddot{\boldsymbol{q}}_\mathrm{d})\boldsymbol{s}\,. \quad (8.100)$$

It is worth noting that the desired compensation is adopted in both the control and adaptive laws such that the computational load can be drastically reduced. For the sake of this analysis, we note that

$$\begin{aligned}&\|\mathbf{Y}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}})\boldsymbol{a} - \mathbf{Y}(\boldsymbol{q}_\mathrm{d}, \dot{\boldsymbol{q}}_\mathrm{d}, \ddot{\boldsymbol{q}}_\mathrm{d})\hat{\boldsymbol{a}}\| \\ &\quad \le \zeta_1\|\boldsymbol{e}_\mathrm{q}\| + \zeta_2\|\boldsymbol{e}_\mathrm{q}\|^2 + \zeta_3\|\boldsymbol{s}\| + \zeta_4\|\boldsymbol{s}\|\|\boldsymbol{e}_\mathrm{q}\|\,,\end{aligned}$$

where ζ_1, ζ_2, ζ_3, and ζ_4 are positive constants. In order to achieve trajectory tracking, it is required that

$$\begin{aligned}k_\mathrm{a} &> \zeta_2 + \zeta_4\,,\\ k_\mathrm{p} &> \frac{\zeta_1}{2} + \frac{\zeta_2}{4}\,,\\ k_\mathrm{v} &> \frac{\zeta_1}{2} + \zeta_3 + \frac{\zeta_2}{4}\,,\end{aligned}$$

(i. e., the gains k_a, k_p, and k_v must be sufficiently large). The convergence of the tracking error to zero with boundedness on all internal signals can be proved through application of Lyapunov stability theory with the following Lyapunov-like function

$$V = \frac{1}{2}\boldsymbol{s}^\mathrm{T}\mathbf{H}(\boldsymbol{q})\boldsymbol{s} + \frac{1}{2}k_\mathrm{p}\boldsymbol{e}_\mathrm{q}^\mathrm{T}\boldsymbol{e}_\mathrm{q} + \frac{1}{2}\tilde{\boldsymbol{a}}^\mathrm{T}\boldsymbol{\Gamma}^{-1}\tilde{\boldsymbol{a}}\,, \quad (8.101)$$

whose time derivative along the trajectories of the closed-loop system can be derived as

$$\dot{V} \le -\boldsymbol{x}^\mathrm{T}\mathbf{Q}\boldsymbol{x}\,, \quad (8.102)$$

where

$$\boldsymbol{x} = \begin{pmatrix}\|\boldsymbol{e}_\mathrm{q}\| \\ \|\boldsymbol{s}\|\end{pmatrix}\,,$$

$$\mathbf{Q} = \begin{pmatrix}\dfrac{k_\mathrm{p} - \zeta_2}{4} & \dfrac{-\zeta_1}{2} \\ \dfrac{-\zeta_1}{2} & \dfrac{k_\mathrm{v} - \zeta_3 - \zeta_4}{4}\end{pmatrix}\,.$$

Detailed stability analysis can be found in [8.45].

8.7.5 Summary

Since the computed-torque control suffers from parametric uncertainty, a variety of adaptive control schemes have been proposed. Firstly, we have presented an adaptive control scheme based on computed-torque control. Then, in order to overcome the mentioned drawbacks such as the measurability of the joint acceleration and the invertibility of the estimated inertia matrix, we presented an alternative adaptive control scheme that is free of these drawbacks. Recently, to incorporate a physics viewpoint into control, adaptive passivity-based control has become popular, and hence is introduced and discussed here. Finally, to reduce the computational load of the adaptive schemes, we presented an adaptive control with desired compensation.

Further Reading

A computationally very fast scheme dealing with adaptive control of rigid manipulators was presented in [8.46]. The stability analysis was completed by assuming that the joint dynamics are decoupled, i. e., that each joint is considered as an independent second-order linear system. A decentralized high-order adaptive variable-structure control is discussed in [8.47], the proposed scheme makes both position and velocity tracking errors of robot manipulators globally converge to zero asymptotically while allowing all signals in closed-loop systems to be bounded without the information of manipulator parameters. Other pioneering works in the field can be found, for example, in [8.48, 49]; although none of the fundamental dynamic model properties are used, the complete dynamics are taken into account, but the control input is discontinuous and may lead to chattering. Positive definiteness of the inertia matrix is explicitly used in [8.50], although it was assumed that some time-varying quantities remain constant during the adaptation. It is interesting to note that all of these schemes were based on the concept of model reference adaptive control (MRAC) developed in [8.51] for linear systems. Therefore, they are conceptually very different from the truly nonlinear schemes presented in this section.

Avidbots Ex1016
Page 225 of 2154

A passive-based modified version of the least-squares estimation scheme has been proposed in [8.52] and [8.53], which guaranteed closed-loop stability of the scheme. Other schemes can be found in [8.54], where no use is made of the skew-symmetry property, and in [8.55], where the recursive Newton–Euler formulations is used instead of the Lagrange formulation to derive the manipulator dynamics, and thus computation is simplified to facilitate practical implementation.

Even though adaptive control provides a solution to the problem of parametric uncertainty, the robustness of adaptive controllers remains a topic of great interest in the field. Indeed, measurement noise or unmodeled dynamics (e.g., flexibility) may result in unbounded closed-loop signals. In particular, the estimated parameters may diverge; this is a well-known phenomenon in adaptive control and is called parameter drift. Solutions inspired from the adaptive control of linear systems have been studied [8.56, 57], where a modified estimation ensures boundedness of the estimates. In [8.58], the controller in [8.32] is modified to enhance robustness.

8.8 Optimal and Robust Control

Given a nonlinear system, such as robotic manipulators, one can develop many stabilizing controls [8.29, 41]. In other words, the stability of the control system cannot determine a unique controller. It is natural that one seeks an optimal controller among the many stable ones. However, the design of an optimal controller is possible provided that a rather exact information on the target system is available, such as an exact system model [8.34, 59]. In the presence of discrepancy between the real system and its mathematical model, a designed optimal controller is no longer optimal, and may even end up being instable in the actual system. Generally speaking, the optimal control design framework is not the best one to deal with system uncertainty. To handle system uncertainty from the control design stage, a robust control design framework is necessary [8.60]. One of main objectives of robust control is to keep the controlled system stable even in presence of uncertainties in the mathematical model, unmodeled dynamics, and the like.

Let us consider an affine nonlinear system described by nonlinear time-varying differential equation in the state $\boldsymbol{x} = (x_1, x_2, \cdots, x_n)^{\mathrm{T}} \in R^n$

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}, t) + \mathbf{G}(\boldsymbol{x}, t)\boldsymbol{u} + \mathbf{P}(\boldsymbol{x}, t)\boldsymbol{w} \,, \tag{8.103}$$

where $\boldsymbol{u} \in R^m$ is the control input, and $\boldsymbol{w} \in R^w$ is the disturbance. Without disturbances or unmodeled dynamics, the system simplifies to

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}, t) + \mathbf{G}(\boldsymbol{x}, t)\boldsymbol{u} \,. \tag{8.104}$$

Actually, there are many kinds of methods describing the nonlinear system according to the objective of control [8.1, 16, 21, 23, 34, 54].

8.8.1 Quadratic Optimal Control

Every optimal controller is based on its own cost function [8.61, 62]. One can define a cost function as [8.63, 64]

$$z = \mathbf{H}(\boldsymbol{x}, t)\boldsymbol{x} + \mathbf{K}(\boldsymbol{x}, t)\boldsymbol{u} \,,$$

such that $\mathbf{H}^{\mathrm{T}}(\boldsymbol{x}, t)\mathbf{K}(\boldsymbol{x}, t) = \mathbf{0}$, $\mathbf{K}^{\mathrm{T}}(\boldsymbol{x}, t)\mathbf{K}(\boldsymbol{x}, t) = \mathbf{R}(\boldsymbol{x}, t) > \mathbf{0}$, and $\mathbf{H}^{\mathrm{T}}(\boldsymbol{x}, t)\mathbf{H}(\boldsymbol{x}, t) = \mathbf{Q}(\boldsymbol{x}, t) > \mathbf{0}$. Then, we have

$$\frac{1}{2}z^{\mathrm{T}}z = \frac{1}{2}\boldsymbol{x}^{\mathrm{T}}\mathbf{Q}(\boldsymbol{x}, t)\boldsymbol{x} + \frac{1}{2}\boldsymbol{u}^{\mathrm{T}}\mathbf{R}(\boldsymbol{x}, t)\boldsymbol{u} \,.$$

The quadratic optimal control for the system (8.104) is found by solving, for a first order differentiable positive-definite function $V(\boldsymbol{x}, t)$, the Hamilton–Jacobi–Bellman (HJB) equation [8.34, 59]

$$\begin{aligned} 0 = \mathrm{HJB}(\boldsymbol{x}, t; V) &= V_t(\boldsymbol{x}, t) + \boldsymbol{V}_x(\boldsymbol{x}, t)\boldsymbol{f}(\boldsymbol{x}, t) \\ &\quad - \frac{1}{2}\boldsymbol{V}_x(\boldsymbol{x}, t)\mathbf{G}(\boldsymbol{x}, t)\mathbf{R}^{-1}(\boldsymbol{x}, t)\mathbf{G}^{\mathrm{T}}(\boldsymbol{x}, t)\boldsymbol{V}_x^{\mathrm{T}}(\boldsymbol{x}, t) \\ &\quad + \frac{1}{2}\mathbf{Q}(\boldsymbol{x}, t) \,, \end{aligned}$$

where $V_t = \frac{\partial V}{\partial t}$ and $\boldsymbol{V}_x = \frac{\partial V}{\partial \boldsymbol{x}^{\mathrm{T}}}$. Then the quadratic optimal control is defined by

$$\boldsymbol{u} = -\mathbf{R}^{-1}(\boldsymbol{x}, t)\mathbf{G}^{\mathrm{T}}(\boldsymbol{x}, t)\boldsymbol{V}_x^{\mathrm{T}}(\boldsymbol{x}, t) \,. \tag{8.105}$$

Note that the HJB equation is a nonlinear second-order partial differential equation in $V(\boldsymbol{x}, t)$.

Unlike the aforementioned optimal control problem, the so-called inverse quadratic optimal control problem is to find a set of $\mathbf{Q}(\boldsymbol{x}, t)$ and $\mathbf{R}(\boldsymbol{x}, t)$ for which the HJB equation has a solution $V(\boldsymbol{x}, t)$. Then the inverse quadratic optimal control is defined by (8.105).

8.8.2 Nonlinear $\mathcal{H}_\infty$ Control

When disturbances are not negligible, one can deal with their effect such that

$$\int_0^t z^{\mathrm{T}}(\boldsymbol{x}, \tau)z(\boldsymbol{x}, \tau)\,\mathrm{d}\tau \le \gamma^2 \int_0^t \boldsymbol{w}^{\mathrm{T}}\boldsymbol{w}\,\mathrm{d}\tau \,, \tag{8.106}$$

Avidbots Ex1016
Page 226 of 2154

where $\gamma > 0$ specifies the L_2-gain of the closed-loop system from the disturbance input $\boldsymbol{w}$ to the cost variable $\boldsymbol{z}$. This is called the L_2-gain attenuation requirement [8.63–65]. One systematic way to design an optimal and robust control is given by the nonlinear $\mathcal{H}_\infty$ optimal control. Let $\gamma > 0$ be given, by solving the following equation

$$\begin{aligned}
&\mathrm{HJI}_\gamma(\boldsymbol{x}, t; V) \\
&\quad = V_t(\boldsymbol{x}, t) + V_x(\boldsymbol{x}, t)\boldsymbol{f}(\boldsymbol{x}, t) \\
&\qquad - \frac{1}{2} V_x(\boldsymbol{x}, t)\{\mathbf{G}(\boldsymbol{x}, t)\mathbf{R}^{-1}(\boldsymbol{x}, t)\mathbf{G}^{\mathrm{T}}(\boldsymbol{x}, t) \\
&\qquad - \gamma^{-2}\mathbf{P}(\boldsymbol{x}, t)\mathbf{P}^{\mathrm{T}}(\boldsymbol{x}, t)\}V_x^{\mathrm{T}}(\boldsymbol{x}, t) \\
&\qquad + \frac{1}{2}\mathbf{Q}(\boldsymbol{x}, t) \le 0 \,, \qquad (8.107)
\end{aligned}$$

then the control is defined by

$$\boldsymbol{u} = -\mathbf{R}^{-1}(\boldsymbol{x}, t)\mathbf{G}^{\mathrm{T}}(\boldsymbol{x}, t)V_x^{\mathrm{T}}(\boldsymbol{x}, t) \,. \qquad (8.108)$$

The partial differential inequality (8.107) is called the Hamilton–Jacobi–Isaac (HJI) inequality. Then one can define the inverse nonlinear $\mathcal{H}_\infty$ optimal control problem that finds a set of $\mathbf{Q}(\boldsymbol{x}, t)$ and $\mathbf{R}(\boldsymbol{x}, t)$ such that the L_2-gain requirement is achieved for a prescribed L_2-gain γ [8.66].

Two things deserve further comment. The first is that the L_2-gain requirement is only valid for disturbance signals $\boldsymbol{w}$ whose L_2-norm is bounded. The second is that $\mathcal{H}_\infty$ optimal control is not uniquely defined. Hence, one can choose a quadratic optimal among many $\mathcal{H}_\infty$ optimal controllers. Precisely speaking, the control (8.108) should be called $\mathcal{H}_\infty$ *suboptimal* control, since the desired L_2-gain is prescribed a priori. A true $\mathcal{H}_\infty$ optimal control is to find a minimal value of γ, such that the L_2-gain requirement is achieved.

8.8.3 Passivity-Based Design of Nonlinear $\mathcal{H}_\infty$ Control

There are many methodologies for the design of optimal and/or robust controls. Among these, passivity-based controls can take full advantage of the properties described above [8.31]. They consist of two parts: one coming from the reference motion compensation while preserving the passivity of the system, and the other to achieve stability, robustness, and/or optimality [8.66, 67].

Let us suppose that the dynamic parameters are identified as $\widehat{\mathbf{H}}(\boldsymbol{q})$, $\widehat{\mathbf{C}}(\boldsymbol{q}, \dot{\boldsymbol{q}})$, and $\widehat{\boldsymbol{\tau}}_{\mathrm{g}}(\boldsymbol{q})$, whose counterparts are $\mathbf{H}(\boldsymbol{q})$, $\mathbf{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})$, and $\boldsymbol{\tau}_{\mathrm{g}}(\boldsymbol{q})$, respectively. Then, passivity-based control generates the following tracking control laws

$$\boldsymbol{\tau} = \widehat{\mathbf{H}}(\boldsymbol{q})\ddot{\boldsymbol{q}}_{\mathrm{ref}} + \widehat{\mathbf{C}}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}}_{\mathrm{ref}} + \widehat{\boldsymbol{\tau}}_{\mathrm{g}}(\boldsymbol{q}) - \boldsymbol{u} \,, \qquad (8.109)$$

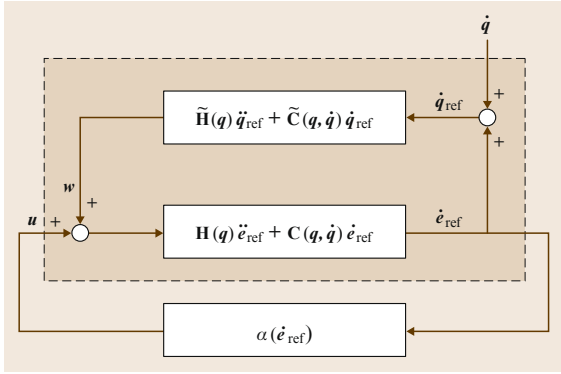


Fig. 8.7 The closed-loop system according to (8.111)

where $\ddot{\boldsymbol{q}}_{\mathrm{ref}}$ is the reference acceleration defined by

$$\ddot{\boldsymbol{q}}_{\mathrm{ref}} = \ddot{\boldsymbol{q}}_{\mathrm{d}} + \mathbf{K}_{\mathrm{V}}\dot{\boldsymbol{e}}_{\mathrm{q}} + \mathbf{K}_{\mathrm{P}}\boldsymbol{e}_{\mathrm{q}} \,, \qquad (8.110)$$

where $\mathbf{K}_{\mathrm{V}} = \mathrm{diag}\{k_{\mathrm{V},i}\} > 0$ and $\mathbf{K}_{\mathrm{P}} = \mathrm{diag}\{k_{\mathrm{P},i}\} > 0$. Two parameters are involved in generating the reference acceleration. Sometimes the following alternative method can be adopted

$$\ddot{\boldsymbol{q}}_{\mathrm{ref}} = \ddot{\boldsymbol{q}}_{\mathrm{d}} + \mathbf{K}_{\mathrm{V}}\dot{\boldsymbol{e}}_{\mathrm{q}} \,.$$

This reduces the order of the closed-loop system because the state $\boldsymbol{x} = (\boldsymbol{e}_{\mathrm{q}}^{\mathrm{T}}, \dot{\boldsymbol{e}}_{\mathrm{q}}^{\mathrm{T}})^{\mathrm{T}}$ is sufficient for the system description, while the definition of (8.110) requires the state $\boldsymbol{x} = (\int \boldsymbol{e}_{\mathrm{q}}^{\mathrm{T}}, \boldsymbol{e}_{\mathrm{q}}^{\mathrm{T}}, \dot{\boldsymbol{e}}_{\mathrm{q}}^{\mathrm{T}})^{\mathrm{T}}$.

In Fig. 8.7, the closed-loop dynamics under the control is given by

$$\mathbf{H}(\boldsymbol{q})\ddot{\boldsymbol{e}}_{\mathrm{ref}} + \mathbf{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{e}}_{\mathrm{ref}} = \boldsymbol{u} + \boldsymbol{w} \,, \qquad (8.111)$$

where

$$\begin{aligned}
\ddot{\boldsymbol{e}}_{\mathrm{ref}} &= \ddot{\boldsymbol{e}}_{\mathrm{q}} + \mathbf{K}_{\mathrm{V}}\dot{\boldsymbol{e}}_{\mathrm{q}} + \mathbf{K}_{\mathrm{P}}\boldsymbol{e}_{\mathrm{q}} \,, \\
\dot{\boldsymbol{e}}_{\mathrm{ref}} &= \dot{\boldsymbol{e}}_{\mathrm{q}} + \mathbf{K}_{\mathrm{V}}\boldsymbol{e}_{\mathrm{q}} + \mathbf{K}_{\mathrm{P}}\int \boldsymbol{e}_{\mathrm{q}} \,.
\end{aligned}$$

If $\boldsymbol{d}(t) = 0$ and $\widehat{\mathbf{H}} = \mathbf{H}$, $\widehat{\mathbf{C}} = \mathbf{C}$, $\widehat{\boldsymbol{\tau}}_{\mathrm{g}} = \boldsymbol{\tau}_{\mathrm{g}}$, then $\boldsymbol{w} = 0$. Otherwise, the disturbance is defined as

$$\boldsymbol{w} = \widetilde{\mathbf{H}}(\boldsymbol{q})\ddot{\boldsymbol{q}}_{\mathrm{ref}} + \widetilde{\mathbf{C}}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}}_{\mathrm{ref}} + \widetilde{\boldsymbol{\tau}}_{\mathrm{g}}(\boldsymbol{q}) + \boldsymbol{d}(t) \,, \qquad (8.112)$$

where $\widetilde{\mathbf{H}} = \mathbf{H} - \widehat{\mathbf{H}}$, $\widetilde{\mathbf{C}} = \mathbf{C} - \widehat{\mathbf{C}}$, and $\widetilde{\boldsymbol{\tau}}_{\mathrm{g}} = \widehat{\boldsymbol{\tau}}_{\mathrm{g}} - \boldsymbol{\tau}_{\mathrm{g}}$. It is of particular interest that the system (8.111) defines a passive mapping between $\boldsymbol{u} + \boldsymbol{w}$ and $\dot{\boldsymbol{e}}_{\mathrm{ref}}$.

According to the manner in which the auxiliary control input $\boldsymbol{u}$ is specified, passivity-based control can achieve stability, robustness, and/or optimality (Fig. 8.7).

Avidbots Ex1016
Page 227 of 2154

8.8.4 A Solution to Inverse Nonlinear $\mathcal{H}_\infty$ Control

Let us define the auxiliary control input by the reference-error feedback

$$\boldsymbol{u} = -\alpha \mathbf{R}^{-1}(\boldsymbol{x}, t)\dot{\boldsymbol{e}}_{\mathrm{ref}} , \tag{8.113}$$

where $\alpha > 1$ is arbitrary. Then, the control provides the inverse nonlinear $\mathcal{H}_\infty$ optimality.

Theorem 8.1 Inverse Nonlinear $\mathcal{H}_\infty$ Optimality [8.66]
Let the reference acceleration generation gain matrices $\mathbf{K}_\mathrm{V}$ and $\mathbf{K}_\mathrm{P}$ satisfy

$$\mathbf{K}_\mathrm{V}^2 > 2\mathbf{K}_\mathrm{P} . \tag{8.114}$$

Then for a given $\gamma > 0$, the reference error feedback

$$\boldsymbol{u} = -\mathbf{K}\dot{\boldsymbol{e}}_{\mathrm{ref}} = -\mathbf{K}\left(\dot{\boldsymbol{e}}_\mathrm{q} + \mathbf{K}_\mathrm{V}\boldsymbol{e}_\mathrm{q} + \mathbf{K}_\mathrm{P}\int \boldsymbol{e}_\mathrm{q}\right) \tag{8.115}$$

satisfies the L_2-gain attenuation requirement for

$$\mathbf{Q} = \begin{pmatrix} \mathbf{K}_\mathrm{P}^2\mathbf{K}_\gamma & 0 & 0 \\ 0 & (\mathbf{K}_\mathrm{V}^2 - 2\mathbf{K}_\mathrm{P})\mathbf{K}_\gamma & 0 \\ 0 & 0 & \mathbf{K}_\gamma \end{pmatrix} , \tag{8.116}$$

$$\mathbf{R} = \mathbf{K}^{-1} , \tag{8.117}$$

provided that

$$\mathbf{K}_\gamma = \mathbf{K} - \frac{1}{\gamma^2}\mathbf{I} > 0 . \tag{8.118}$$

Given γ, one can set $\mathbf{K} = \alpha\frac{1}{\gamma^2}\mathbf{I}$ for $\alpha > 1$. This yields $\mathbf{K}_\gamma = (\alpha - 1)\frac{1}{\gamma^2}\mathbf{I}$.

When the inertia matrix is identified as a diagonal constant matrix such as $\widehat{\mathbf{H}} = \mathrm{diag}\{\widehat{m}_i\}$, one should set $\widehat{\mathbf{C}} = \mathbf{0}$. In addition, one can set $\widehat{\boldsymbol{\tau}}_\mathrm{g} = \mathbf{0}$. Then this results in a decoupled PID control of the form

$$\begin{aligned}\tau_i = {} & \widehat{m}_i\left(\ddot{q}_{\mathrm{d},i} + k_{\mathrm{V},i}\dot{e}_{\mathrm{q},i} + k_{\mathrm{P},i}e_{\mathrm{q},i}\right) \\ & + \alpha\frac{1}{\gamma^2}\left(\dot{e}_{\mathrm{q},i} + k_{\mathrm{V},i}e_{\mathrm{q},i} + k_{\mathrm{P},i}\int e_{\mathrm{q},i}\right)\end{aligned}$$

for $\alpha > 1$, which can be rewritten as

$$\begin{aligned}\tau_i = {} & \widehat{m}_i\ddot{q}_{\mathrm{d},i} + \left(\widehat{m}_i k_{\mathrm{V},i} + \alpha\frac{1}{\gamma^2}\right)\dot{e}_{\mathrm{q},i} \\ & + \left(\widehat{m}_i k_{\mathrm{P},i} + \alpha\frac{k_{\mathrm{V},i}}{\gamma^2}\right)e_{\mathrm{q},i} + \alpha\frac{k_{\mathrm{P},i}}{\gamma^2}\int e_{\mathrm{q},i} .\end{aligned} \tag{8.119}$$

This leads to a PID control with the desired acceleration feedforward [8.68] given by

$$\tau_i = \widehat{m}_i\ddot{q}_{\mathrm{d},i} + k^*_{\mathrm{V},i}\dot{e}_{\mathrm{q},i} + k^*_{\mathrm{P},i}e_{\mathrm{q},i} + k^*_{\mathrm{I},i}\int e_{\mathrm{q},i} , \tag{8.120a}$$

where

$$k^*_{\mathrm{V},i} = \widehat{m}_i k_{\mathrm{V},i} + \alpha\frac{1}{\gamma^2} , \tag{8.120b}$$

$$k^*_{\mathrm{P},i} = \widehat{m}_i k_{\mathrm{P},i} + \alpha\frac{k_{\mathrm{V},i}}{\gamma^2} , \tag{8.120c}$$

$$k^*_{\mathrm{I},i} = \alpha\frac{k_{\mathrm{P},i}}{\gamma^2} . \tag{8.120d}$$

8.9 Trajectory Generation and Planning

This section deals with the problem of *reference trajectory generation*, that is, the computation of desired position, velocity, acceleration and/or force/torque signals that are used as input values for the robot motion controllers introduced in Sects 8.3–8.8.

8.9.1 Geometric Paths and Trajectories

Path Planning

A path is a geometric representation of a plan to move from a start to a target pose. The task of planning is to find a collision-free path among a collection of static and dynamic obstacles. Path planning can also include the consideration of dynamic constraints such as workspace boundaries, maximum velocities, maximum accelerations, and maximum jerks. We distinguish between *online* and *offline* path planning algorithms. Offline planned paths are static and calculated prior to execution. Online methods require algorithms that meet real-time constraints (i. e., algorithms that do not exceed a determinable worst-case computation time) to enable path (re-)calculations and/or adaptations during the robot motions in order to react to and interact with dynamic environments. This means that a robot

Avidbots Ex1016
Page 228 of 2154

moves along a path that has not necessarily been computed completely, and which may change during the movement. Details about path planning concepts are described in Chap. 7, in Parts D and E, and specifically in Chap. 47.

Trajectory Planning

A trajectory is more than a path: It also includes velocities, accelerations, and/or jerks along a path (VIDEO 760). A common method is computing trajectories for a priori specified paths, which fulfill a certain criterion (e.g., minimum execution time). We distinguish between *online* and *offline* trajectory planning methods. An offline calculated trajectory cannot be influenced during its execution, while online trajectory planning methods can (re-)calculate and/or adapt robot motions behavior during the movement. The reasons for this (re-)calculation and/or adaptation can vary: improvement of accuracy, better utilization of currently available dynamics, reaction to and interaction with a dynamic environment, or reaction to (sensor) events. Besides the distinction between online and offline methods, we can further distinguish between (1) one-dimensional (1-D) and multi-dimensional trajectories and (2) single-point and multi-point trajectories. Multi-point trajectories typically relate to a path.

8.9.2 Joint Space and Operational Space Trajectories

Depending on the control state space, trajectory generators provide set points for tracking controllers in joint space or in operational space. In either space, a trajectory can be represented in several different ways: cubic splines, quintic splines, higher-order splines, harmonics (sinusoidal functions), exponential functions, Fourier series, and more.

Joint Space Trajectories

Consider the torque control input (8.29)

$$\boldsymbol{\tau} = \mathbf{H}(\boldsymbol{q})\boldsymbol{v} + \mathbf{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{\tau}_{\mathrm{g}}(\boldsymbol{q}) \,,$$

and the PD controller (8.30)

$$\boldsymbol{v} = \ddot{\boldsymbol{q}}_{\mathrm{d}} + \mathbf{K}_{\mathrm{V}}(\dot{\boldsymbol{q}}_{\mathrm{d}} - \dot{\boldsymbol{q}}) + \mathbf{K}_{\mathrm{P}}(\boldsymbol{q}_{\mathrm{d}} - \boldsymbol{q}) \,, \tag{8.121}$$

or PID controller (8.31), respectively, the task of a trajectory generator in joint space coordinates is computing the signals $\boldsymbol{q}_{\mathrm{d}}(t)$, $\dot{\boldsymbol{q}}_{\mathrm{d}}(t)$, and $\ddot{\boldsymbol{q}}_{\mathrm{d}}(t)$. These three signals contain the reference trajectory and are used as input values for the tracking controller.

During nominal operation the joint torques required to execute a trajectory should not exceed joint force/torque limits $\boldsymbol{\tau}_{\min}(t)$ and $\boldsymbol{\tau}_{\max}(t)$,

$$\boldsymbol{\tau}_{\min}(t) \leq \boldsymbol{\tau}(t) \leq \boldsymbol{\tau}_{\max}(t) \;\forall\; t \in \mathbb{R}. \tag{8.122}$$

Operational Space Trajectories

Similarly to (8.29), we can also consider trajectories represented by $\boldsymbol{x}_{\mathrm{d}}$, $\dot{\boldsymbol{x}}_{\mathrm{d}}$, and $\ddot{\boldsymbol{x}}_{\mathrm{d}}$ for an operational space controller (8.9)

$$\boldsymbol{f}_{\mathrm{c}} = \boldsymbol{\Lambda}(\boldsymbol{q})\boldsymbol{\mu} + \boldsymbol{\Gamma}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{x}} + \boldsymbol{\eta}(\boldsymbol{q}) \,,$$

with a PD control law of

$$\boldsymbol{\mu} = \ddot{\boldsymbol{x}}_{\mathrm{d}} + \mathbf{K}_{\mathrm{V}}(\dot{\boldsymbol{x}}_{\mathrm{d}} - \dot{\boldsymbol{x}}) + \mathbf{K}_{\mathrm{P}}(\boldsymbol{x}_{\mathrm{d}} - \boldsymbol{x}) \,. \tag{8.123}$$

With the transformation into joint space using the inverse of (8.8), the operational space trajectory generator must assure that the limits given in (8.122) are not violated. It is the responsibility of the path planner (Chap. 7) that all points along the trajectory are in the robot workspace and that start and goal poses can be reached in the same joint configuration. It is the responsibility of the trajectory planner that joint torque and velocity constraints are not violated even in the presence of kinematic singularities.

8.9.3 Trajectory Representations

Mathematical Representations

Functions for $\ddot{\boldsymbol{q}}_{\mathrm{d}}(t)$ (8.121) and $\ddot{\boldsymbol{x}}_{\mathrm{d}}$ (8.123) can be represented in several ways that are described here.

Polynomial Trajectories. One of the simplest ways to represent a robot trajectory is a polynomial function of degree m for each joint $i \in \{1, \ldots, n\}$

$$q_i(t) = a_{i,0} + a_{i,1}t + a_{i,2}t^2 + \ldots + a_{i,m}t^m \,, \tag{8.124}$$

so that $\boldsymbol{q}(t)$ can be composed (or $\boldsymbol{x}(t)$ in operational space, respectively). In the simplest case, cubic polynomials are used, which, however, leads to non-steady acceleration signals with infinite jerks. Quintic and higher-order polynomials allow for steady acceleration signals as well as arbitrary position, velocity, and acceleration vectors and the beginning and at the end of the trajectory. To determine the coefficients $a_{i,j}\ \forall\ (i,j) \in \{1, \ldots, n\} \times \{0, \ldots, m\}$ of (8.124), the execution time t_{trgt}, at which the target state will be reached needs to be known. The left part of Fig. 8.8a shows a quintic trajectory for three DOFs starting at $t_0 = 0\,s$ with an execution time of $t_{\mathrm{trgt}} = 2.5\,s$. To connect a trajectory segment to preceding and succeeding segments, the fol-

Avidbots Ex1016
Page 229 of 2154

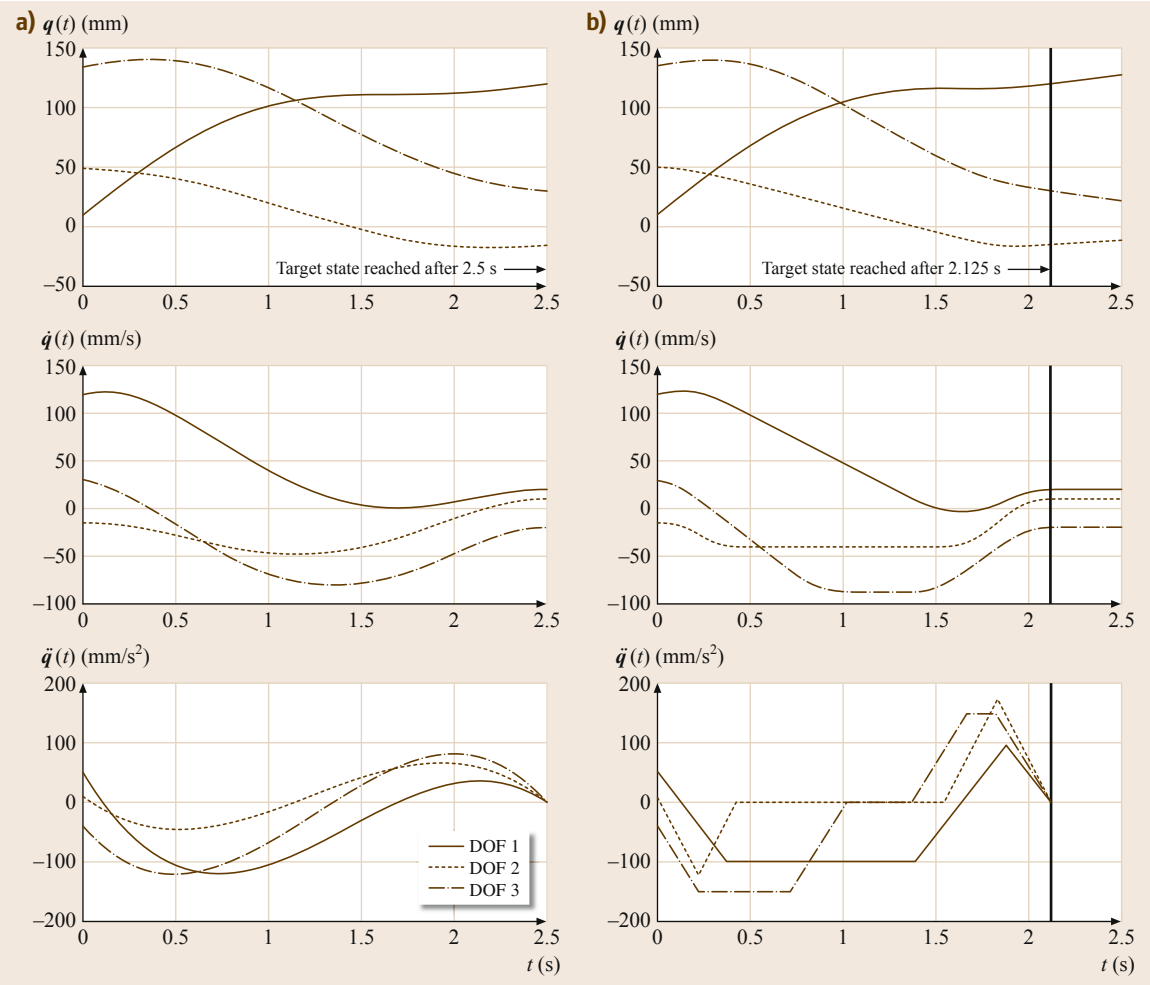


Fig.8.8a,b Two sample trajectories for a three-DOF robot ($n = 3$). The trajectory in **(a)** is represented by quintic splines, and the trajectory in **(b)** by piecewise polynomials

lowing six constraints need to be satisfied for all n joints

$$
\begin{aligned}
q_i(t_0) &= q_{i,\mathrm{start}} = a_{i,0} \\
q_i(t_{\mathrm{trgt}}) &= q_{i,\mathrm{trgt}} = a_{i,0} + a_{i,1}t_{\mathrm{trgt}} + a_{i,2}t_{\mathrm{trgt}}^2 \\
&\quad + a_{i,3}t_{\mathrm{trgt}}^3 + a_{i,4}t_{\mathrm{trgt}}^4 + a_{i,5}t_{\mathrm{trgt}}^5 \\
\dot{q}_i(t_0) &= \dot{q}_{i,0} = a_{i,1} \\
\dot{q}_i(t_{\mathrm{trgt}}) &= \dot{q}_{i,\mathrm{trgt}} = a_{i,1} + 2a_{i,2}t_{\mathrm{trgt}} + 3a_{i,3}t_{\mathrm{trgt}}^2 \\
&\quad + 4a_{i,4}t_{\mathrm{trgt}}^3 + 5a_{i,5}t_{\mathrm{trgt}}^4 \\
\ddot{q}_i(t_0) &= \ddot{q}_{i,0} = 2a_{i,2} \\
\ddot{q}_i(t_{\mathrm{trgt}}) &= \ddot{q}_{i,\mathrm{trgt}} = 2a_{i,2} + 6a_{i,3}t_{\mathrm{trgt}} + 12a_{i,4}t_{\mathrm{trgt}}^2 \\
&\quad + 20a_{i,5}t_{\mathrm{trgt}}^3 \,.
\end{aligned}
$$

A unique closed-form solution can be computed, so that all polynomial coefficients $a_{i,j}\ \forall\ (i,j) \in \{1, \ldots, n\} \times \{0, \ldots, 5\}$ can be determined for one trajectory segment.

Piecewise Polynomials. Polynomials of different degrees can be concatenated to represent a trajectory between an initial state ($\boldsymbol{q}_0$, $\dot{\boldsymbol{q}}_0$, $\ddot{\boldsymbol{q}}_0$) and a target state ($\boldsymbol{q}_{\mathrm{trgt}}$, $\dot{\boldsymbol{q}}_{\mathrm{trgt}}$, $\ddot{\boldsymbol{q}}_{\mathrm{trgt}}$). For instance, the classical double-S velocity profile [8.69] with trapezoidal acceleration and deceleration profiles and a cruise-velocity segment in the middle consists of seven polynomials of degrees 3-2-3-1-3-2-3 (m in (8.124)). The right part of Fig. 8.8 shows a trajectory represented by piecewise polynomials. To compute time-optimal trajectories under purely kinematic constraints (e.g., $\dot{\boldsymbol{q}}_{\max}$, $\ddot{\boldsymbol{q}}_{\max}$, $\dddot{\boldsymbol{q}}_{\max}$, etc.), piecewise polynomials are used, because they allow always using one signal at its kinematic limit (Fig. 8.8b and [8.70]).

Trigonometric Trajectories. Similarly to (8.124), trigonometric functions can be used to represent harmonic, cycloidal, and elliptic trajectories [8.71, 72]. A simple example for a harmonic trajectory for one joint i is

$$q_i(t) = \frac{q_{i,\mathrm{trgt}} - q_{i,0}}{2}\left(1 - \cos\frac{\pi\,(t - t_0)}{t_{\mathrm{trgt}} - t_0}\right) + q_{i,0}\,. \tag{8.125}$$

While any order of derivatives of trigonometric functions is continuous, they might be discontinuous at t_0 and t_{trgt}.

Other Representations. *Exponential Trajectories* and *Fourier Series Expansions* [8.72] are particularly suited to minimize natural vibrations on robot mechanisms introduced by reference trajectories.

Trajectories and Paths

To draw the connection to Chap. 7 and Parts D and E, trajectories and paths are often tightly coupled.

Trajectories Along Predefined Paths. A path in joint space can be described by a function $\boldsymbol{q}(s(t))$ with $s \in [t_0, t_{\mathrm{trgt}}]$, where the start configuration of the path is $\boldsymbol{q}(t_0)$ and the target configuration is $\boldsymbol{q}(t_{\mathrm{trgt}})$. To move a robot along the path, an appropriate function $s(t)$ needs to be computed that does not violate any of the kinematic and dynamic constraints [8.73–75]. If a path is given in operational space, $\boldsymbol{x}(s(t))$ can be mapped to $\boldsymbol{q}(t)$ (Sect. 8.2).

Multi-Dimensional Trajectories. Instead of using a one-dimensional function $s(t)$ to parameterize a path segment, trajectories can also be described by individual functions for each DOF i to represent $\boldsymbol{q}(s(t))$ or $\boldsymbol{x}(s(t))$, respectively. To connect two arbitrary states, the signals for each individual degree of freedom need to be time-synchronized [8.70], so that all DOFs reach their target state of motion at the very same instant. Those trajectories may also be phase-synchronized [8.76], so that the trajectories of all DOFs are derived from a one master DOF and only scaled by a factor to achieve homothety [8.77]. The two trajectories in Fig. 8.8 are time-synchronized but not phase-synchronized.

Multi-Point Trajectories. If instead of an entirely defined geometric path or a motion to a single way point, an entire series of geometric way points is given, the trajectory that connects all way points in a given state space needs to be computed. Trajectory segments between two way points can be represented with any of the above mentioned representations as long as the position signal and its derivatives up an appropriate order are continuous (at least C^1 continuous). Splines, B-splines, or Bezier splines are used to generate either a reference trajectory or a geometric path, which then can be parameterized with a function $s(t)$.

8.9.4 Trajectory Planning Algorithms

The following part provides an overview of online and offline trajectory planning concepts.

Constraints

Constraints for trajectory planners can manifold:

- *Kinematic*: maximum velocities, accelerations, jerks, etc. and workspace space limits
- *Dynamic*: maximum joint or actuator forces and/or torques
- *Geometric*: no collisions with static and dynamic objects in the workspace
- *Temporal*: reaching a state within a given time interval or at a given time.

These and other constraints may have to be taken into account at the same time. Depending on the robot and the task, additional optimization criteria may be considered (e.g., time-optimality, minimum-jerk, maximum distance to workspace boundaries, minimum energy).

Offline Trajectory Planning

Kahn and *Roth* [8.78] showed results using optimal, linear control theory to achieve a near-time-optimal solution for linearized manipulators. The resulting trajectories are jerk-limited and lead to smaller trajectory-following errors and to less excitation of structural natural frequencies in the system.

The work of *Brady* [8.79] introduced several techniques of trajectory planning in joint space and *Paul* [8.80] and *Taylor* [8.81] published works about the planning of trajectories in Cartesian space in parallel to Brady. *Lin* et al. [8.82] published another purely kinematic approach as did *Castain* and *Paul* [8.69].

Hollerbach [8.83] first introduced the consideration of the nonlinear inverse robot dynamics for the generation of manipulator trajectories.

During the middle of the 1980s, three groups developed techniques for time-optimal trajectory planning for arbitrarily specified paths: *Bobrow* [8.73], *Shin* and *McKay* [8.74], and *Pfeiffer* and *Johanni* [8.75]. Trajectories are composed of three curves: the maximum acceleration curve, the maximum velocity curve, and

Avidbots Ex1016
Page 231 of 2154

the maximum deceleration curve. The proposed algorithms find the intersection points of these three curves.

These algorithms have become the fundament for many follow-up works: *Kyriakopoulos* and *Sridis* [8.84] added minimum-jerk criteria; *Slotine* and *Yang* abandoned the computationally expensive calculation of the maximum velocity curve [8.85]; *Shiller* and *Lu* added methods for handling dynamic singularities [8.86]; *Fiorini* and *Shiller* extended the algorithm for known dynamic environments with moving obstacles [8.87].

Online Trajectory Planning

An online modification of a planned trajectory may have several reasons: (i) The trajectory becomes adapted in order to improve the accuracy with a path specified beforehand; (ii) The robotic system reacts on sensor signals and/or events that cannot be predicted beforehand, because the robot acts in a (partly) unknown and dynamic environment.

Improving Path Accuracy. All previously described off-line trajectory planning methods assume a dynamic model that describes the behavior of the real robot exactly. In practice, this is often not the case, and some robot parameters are only estimated, some dynamic effects remain unmodeled, and system parameters may change during operation. If this is the case, the resulting robot motion is not time-optimal anymore and/or the maximum actuator forces and/or torques are exceeded, which leads to an undesired difference between the specified and the executed path.

Dahl and *Nielsen* [8.88] extended [8.73–75] by adapting the acceleration along the path, so that the underlying trajectory-following controller becomes adapted depending on the current state of motion. The approaches of *Cao* et al. [8.89, 90] use cubic splines to generate smooth paths in joint space with time-optimal trajectories. *Constantinescu* and *Croft* [8.91] suggest a further improvement to the approach of [8.86] with objective to limit the derivative of actuator forces/torques. *Macfarlane* and *Croft* extended this approach further by adding jerk limitations to quintic splines (Fig. 8.8) [8.92].

Sensor-Based Trajectory Adaptation

The last paragraph presented an overview of online trajectory generation methods for improving the path accuracy, while this one focuses on the online consideration of sensor signals, for instance, for the purpose of collision avoidance (VIDEO 757, VIDEO 758) or switching between controllers or control gains (VIDEO 759, VIDEO 761).

In 1988, *Andersson* presented an online trajectory planning for a Ping-Pong-playing PUMA 260 manipulator that computes parameterized quintic polynomials [8.93, 94]. Based on [8.73–75], *Lloyd* and *Hayward* proposed a technique to transition between two different trajectory segments [8.95] using a transition window [8.81]. *Ahn* et al. introduced a method to connect two arbitrary motion states online, which does not take into account kinematic or dynamic constraints [8.96]. *Broquère* et al., *Haschke* et al., and *Kröger* extended this approach for multi-dimensional trajectories, so that kinematic constraints are taken into account [8.70, 97, 98].

Further Reading

Overviews of the domain of robot reference trajectory generation can found in the textbooks of *Biagiotti* and *Melchiorri* [8.72], *Craig* [8.99], *Fu* et al. [8.100], and *Spong* et al. [8.101].

8.10 Digital Implementation

Most controllers introduced in the previous sections are digitally implemented on microprocessors. In this section basic but essential practical issues related to their computer implementation are discussed. When the controller is implemented on a computer control system, the analog inputs are read and the outputs are set with a certain sampling period. This is a drawback compared to analog implementations, since sampling introduces time delays into the control loop. Figure 8.9 shows the overall block diagram of control system with a boxed digital implementation part. When a digital computer is used to implement a control law, it is convenient to divide coding sequence in the interrupt routine into four process routines, as shown in Fig. 8.10. Reading the input signal from sensors and writing the control signal to digital-to-analog (D/A) converters synchronized at the correct frequency is very important. Therefore, these processes are located in the first routine. After saving counter values and extracting D/A values, which are already calculated one step before, the next routine produces reference values. Control routines with filters follow and produce scalar or vector control outputs. Finally, the user interface for checking parameter values is made and will be used for tuning and debugging.

Avidbots Ex1016
Page 232 of 2154

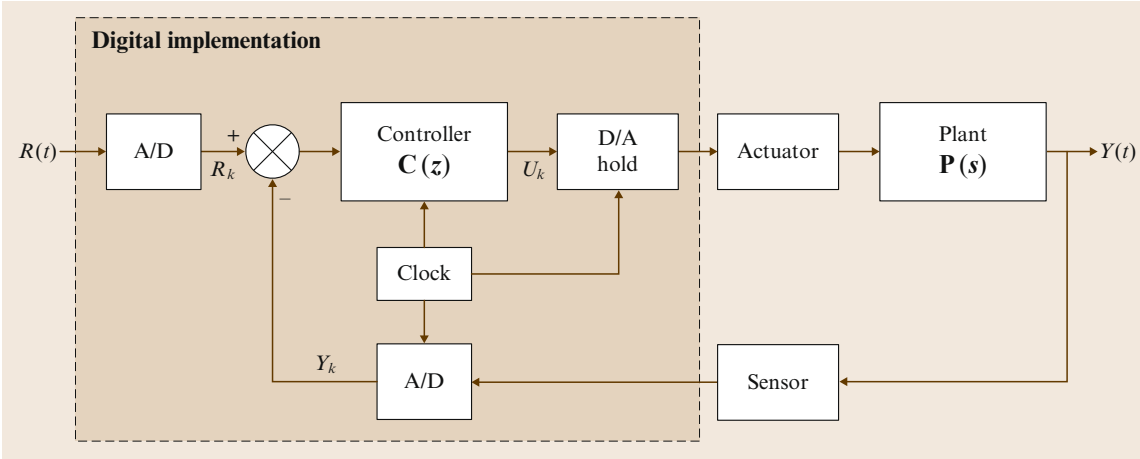


Fig. 8.9 Digital implementation of system control

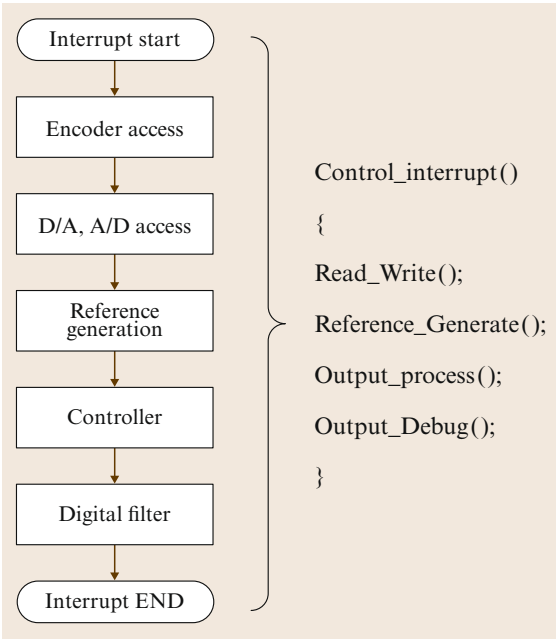


Fig. 8.10 The sequence in the interrupt routine for digital control

8.10.1 Z-Transform for Motion Control Implementation

Continuous-time systems are transformed into discrete-time systems by using the Z-transform. A discrete-time system is used to obtain a mathematical model that gives the behavior of a physical process at the sampling points, although the physical process is still a continuous-time system. A Laplace transform is used for the analysis of control system in the s-domain. In most cases, the design of controllers and filters are done using tools in the s-domain. In order to realize those results in program code, understanding the Z-transform is essential. All controllers and filters designed in the s-domain can be easily translated to a program code through a Z-transform because it has the form of digitized difference sequences.

A PID controller is used as an example. In transfer function form, this controller has the basic structure

$$\frac{Y(s)}{E(s)} = K_\mathrm{P} + \frac{K_\mathrm{I}}{s} + sK_\mathrm{V} \,. \tag{8.126}$$

There are several methods for transformation from the frequency domain to the discrete domain. For stability conservation, backward Euler and Tustin algorithms are often used. Though the Tustin algorithm is known as the more exact one, the backward Euler algorithm is utilized in the following procedure.

After substituting the backward Euler equation into (8.126),

$$s \cong \frac{1-z^{-1}}{T} \,,$$

the following discrete form is produced

$$\frac{Y(z)}{E(z)} = \frac{\alpha + \beta z^{-1} + \gamma z^{-2}}{T(1-z^{-1})} \,, \tag{8.127}$$

where

$$\alpha = K_\mathrm{I}T^2 + K_\mathrm{P}T + K_\mathrm{V} \,,$$
$$\beta = -K_\mathrm{P}T - 2K_\mathrm{V} \,,$$
$$\gamma = K_\mathrm{V} \,.$$

Sometimes a differentiator s in the PID controller makes the implementation infeasible when the mea-

Avidbots Ex1016
Page 233 of 2154

surement noise is severe. One can remedy the controller (8.126) by attaching a lowpass filter with filter time constant σ

$$\frac{Y(s)}{E(s)} = K_\mathrm{P} + \frac{K_\mathrm{I}}{s} + \frac{s}{\sigma s + 1} K_\mathrm{V} \,. \tag{8.128}$$

Again, substituting the backward Euler equation into (8.128) produces

$$\frac{Y(z)}{E(z)} = \frac{\alpha + \beta z^{-1} + \gamma z^{-2}}{1 - \delta z^{-1} - \psi z^{-2}} \,, \tag{8.129}$$

where

$$\begin{aligned}
\alpha &= K_\mathrm{P} + K_\mathrm{I} T + \frac{K_\mathrm{V}}{\sigma + T} \,, \\
\beta &= -\frac{(2\sigma + T)K_\mathrm{P} + \sigma T K_\mathrm{I} + 2K_\mathrm{V}}{\sigma + T} \,, \\
\gamma &= \frac{\sigma K_\mathrm{P} + K_\mathrm{V}}{\sigma + T} \,, \\
\delta &= \frac{2\sigma + T}{\sigma + T} \,, \\
\psi &= -\frac{\sigma}{\sigma + T} \,,
\end{aligned}$$

in which the filter time constant σ is determined from a cutoff frequency f_c[Hz] for removing noises such as $\sigma = \frac{1}{2\pi f_\mathrm{c}}$.

8.10.2 Digital Control for Coding

Inverse Z-transform produces a difference equation for digital control. Furthermore the difference equation can be directly converted into the control program code. Since the inverse Z-transform of $Y(z)$ is y_k and z^{-1} implies the previous sample time, $z^{-1}Y(z) = y_{k-1}$ and $z^{-2}Y(z) = y_{k-2}$.

Now, the PID controller expressed by (8.127) is rearranged using the difference equation

$$\begin{aligned}
T(y_k - y_{k-1}) &= \alpha e_k + \beta e_{k-1} + \gamma e_{k-2} \,, \\
y_k - y_{k-1} &= \frac{1}{T}(\alpha e_k + \beta e_{k-1} + \gamma e_{k-2}) \,.
\end{aligned} \tag{8.130}$$

For pratical use, the PID controller can be directly coded in the program as follows

$$y_k = K_{\mathrm{P,c}} e_k + K_{\mathrm{V,c}} e_k^v + K_{\mathrm{I,c}} e_k^i \,,$$

where

$$\begin{aligned}
e_k &= p_{k,\mathrm{desired}} - p_k \,, \\
e_k^v &= v_{k,\mathrm{desired}} - v_k \\
&= (p_{k,\mathrm{desired}} - p_{k-1,\mathrm{desired}}) - (p_k - p_{k-1}) \\
&= e_k - e_{k-1} \,, \\
e_k^i &= e_{k-1}^i + e_k = \sum_{j=0}^{k} e_k \,,
\end{aligned}$$

in which p_k is the present position, v_k is the present velocity, desired means reference to be followed, and c means coded form for digitial control. Now let us obtain the difference between the present control output and the previous one

$$\begin{aligned}
&y_k - y_{k-1} \\
&\quad = [K_{\mathrm{P,c}} e_k - K_{\mathrm{P,c}} e_{k-1}] \\
&\qquad + [K_{\mathrm{V,c}}(e_k - e_{k-1}) - K_{\mathrm{V,c}}(e_{k-1} - e_{k-2})] \\
&\qquad + [K_{\mathrm{I,c}} e_k^i - K_{\mathrm{I,c}} e_{k-1}^i] \\
&\quad = (K_{\mathrm{P,c}} + K_{\mathrm{V,c}} + K_{\mathrm{I,c}}) e_k \\
&\qquad - (K_{\mathrm{P,c}} + 2K_{\mathrm{V,c}}) e_{k-1} + K_{\mathrm{V,c}} e_{k-2} \,.
\end{aligned} \tag{8.131}$$

Comparing the parameters in (8.130) and (8.131), one obtains

$$\begin{aligned}
\frac{\alpha}{T} &= K_\mathrm{P} + \frac{K_\mathrm{V}}{T} + K_\mathrm{I} T = (K_{\mathrm{P,c}} + K_{\mathrm{V,c}} + K_{\mathrm{I,c}}) \,, \\
\frac{\beta}{T} &= -K_\mathrm{P} - \frac{2K_\mathrm{V}}{T} = -(K_{\mathrm{P,c}} + 2K_{\mathrm{V,c}}) \,, \\
\frac{\gamma}{T} &= \frac{K_\mathrm{V}}{T} = K_{\mathrm{V,c}} \,,
\end{aligned}$$

which shows that there is a relation between the designed and coded forms of the gains

$$\begin{aligned}
K_{\mathrm{P,c}} &= K_\mathrm{P} \,, \\
K_{\mathrm{V,c}} &= \frac{K_\mathrm{V}}{T} \,, \\
K_{\mathrm{I,c}} &= K_\mathrm{I} T \,.
\end{aligned} \tag{8.132}$$

As the sampling frequency is increased in the same system, the coded K_V gain should be increased and the coded K_I gain should be decreased. Using this method, the designed controller can be coded in a digital signal processor (DSP) or microprocessor. However, sufficient analysis and simulation for control algorithms should be performed beforehand to obtain successful control system performance.

In addition, the PID controller with lowpass filter (8.129) can be implemented as

$$\begin{aligned}
y_k - \delta y_{k-1} - \psi y_{k-2} &= \alpha e_k + \beta e_{k-1} + \gamma e_{k-2} \,, \\
y_k &= \delta y_{k-1} + \psi y_{k-2} + \alpha e_k + \beta e_{k-1} + \gamma e_{k-2} \,.
\end{aligned} \tag{8.133}$$

Using the same procedures, one can arrive at the similar control program code for digital control.

Avidbots Ex1016
Page 234 of 2154

PID Control Experiment (Multimedia)
According as the gains change, the performance variations of PID controller implemented in the digital control system are shown in the multimedia source to help readers' understanding VIDEO 25.

8.11 Learning Control

Since many robotic applications, such as pick-and-place operations, paintings, and circuit-board assembly, involve repetitive motions, it is natural to consider the use of data gathered in previous cycles to try to improve the performance of the manipulator in subsequent cycles. This is the basic idea of repetitive control or learning control. Consider the robot model given in Sect. 8.1 and suppose that one is given a desired joint trajectory $\boldsymbol{q}_{\mathrm{d}}(t)$ on a finite time interval $0 \le t \le T$. The reference trajectory $\boldsymbol{q}_{\mathrm{d}}$ is used in repeated trails of the manipulator, assuming either that the trajectory is periodic, $\boldsymbol{q}_{\mathrm{d}}(T) = \boldsymbol{q}_{\mathrm{d}}(0)$, (repetitive control) or that the robot is reinitialized to lie on the desired trajectory at the beginning of each trail (learning control). Hereafter, we use the term learning control to mean either repetitive or learning control.

8.11.1 Pure P-Type Learning Control

Let $\boldsymbol{\tau}_k$ be the input torque during the k-th cycle, which produces an output $\boldsymbol{q}_k(t), 0 \le t \le T_{\mathrm{bnd}}$. Now, let us consider the following set of assumptions:

- Assumption 1: Every trial ends at a fixed time of duration $T_{\mathrm{bnd}} > 0$.
- Assumption 2: Repetition of the initial setting is satisfied.
- Assumption 3: Invariance of the system dynamics is ensured throughout these repeated trails.
- Assumption 4: Every output $\boldsymbol{q}_k$ can be measured and thus the error signal $\Delta\boldsymbol{q}_k = \boldsymbol{q}_k - \boldsymbol{q}_{\mathrm{d}}$ can be utilized in the construction of the next input $\boldsymbol{\tau}_{k+1}$.
- Assumption 5: The dynamics of the robot manipulators is invertible.

The learning control problem is to determine a recursive learning law $\boldsymbol{L}$

$$\boldsymbol{\tau}_{k+1} = \boldsymbol{L}[\boldsymbol{\tau}_k(t), \Delta\boldsymbol{q}_k(t)] , \quad 0 \le t \le T_{\mathrm{bnd}} , \tag{8.134}$$

where $\Delta\boldsymbol{q}_k(t) = \boldsymbol{q}_k(t) - \boldsymbol{q}_{\mathrm{d}}(t)$, such that $\|\Delta\boldsymbol{q}_k\| \to 0$ as $k \to \infty$ in some suitably defined function norm, $\|\cdot\|$. The initial control input can be any control input that produces a stable output, such as PD control. Such learning control schemes are attractive because accurate models of the dynamics need not be known a priori.

Several approaches have been used to generate a suitable learning law $\boldsymbol{L}$ and to prove convergence of the output error. A pure P-type learning law is one of the form

$$\boldsymbol{\tau}_{k+1}(t) = \boldsymbol{\tau}_k(t) - \boldsymbol{\Phi}\Delta\boldsymbol{q}_k(t) , \tag{8.135}$$

and is given this name because the correction term for the input torque at each iteration is proportional to the error $\Delta\boldsymbol{q}_k$. Now let $\boldsymbol{\tau}_d$ be defined by the computed-torque control, i. e.,

$$\boldsymbol{\tau}_d(t) = \mathbf{H}[\boldsymbol{q}_{\mathrm{d}}(t)]\ddot{\boldsymbol{q}}_{\mathrm{d}}(t) + \mathbf{C}[\boldsymbol{q}_{\mathrm{d}}(t), \dot{\boldsymbol{q}}_{\mathrm{d}}(t)]\dot{\boldsymbol{q}}_{\mathrm{d}}(t) + \boldsymbol{\tau}_{\mathrm{g}}[\boldsymbol{q}_{\mathrm{d}}(t)] . \tag{8.136}$$

One should recall that the function $\boldsymbol{\tau}_k$ actually does not need to be computed; it is sufficient to know that it exists. Considering the P-type learning control law, we have

$$\Delta\boldsymbol{\tau}_{k+1}(t) = \Delta\boldsymbol{\tau}_k(t) - \boldsymbol{\Phi}\Delta\boldsymbol{q}_k(t) , \tag{8.137}$$

where $\Delta\boldsymbol{\tau}_k(t) = \boldsymbol{\tau}_k(t) - \boldsymbol{\tau}_d(t)$, so that

$$\|\Delta\boldsymbol{\tau}_{k+1}(t)\|^2 \le \|\Delta\boldsymbol{\tau}_k(t)\|^2 - \beta\|\boldsymbol{\Phi}\Delta\boldsymbol{q}_k(t)\|^2 \tag{8.138}$$

provided there exist positive constant λ and β such that

$$\int_0^{T_{\mathrm{bnd}}} \mathrm{e}^{-\lambda t}\Delta\boldsymbol{q}_k{}^{\mathrm{T}}\Delta\boldsymbol{\tau}_k(t)\mathrm{d}t \ge \frac{1+\beta}{2}\|\boldsymbol{\Phi}\Delta\boldsymbol{q}_k(t)\|^2 \tag{8.139}$$

for all k. It then follows from the inequality above that $\Delta q_k \to 0$ in the norm sense as $k \to \infty$. Detailed stability analysis of this control scheme is given in [8.102, 103].

8.11.2 P-Type Learning Control with a Forgetting Factor

Although pure P-type learning control achieves the desired goal, several strict assumptions may be not valid in actual implementations, for example, there may be an initial setting error. Furthermore, there may be small but

Avidbots Ex1016
Page 235 of 2154

nonrepeatable fluctuations of dynamics. Finally, there may exit a (bounded) measurement noise $\boldsymbol{\xi}_k$ such that

$$\Delta \boldsymbol{q}_k(t) + \boldsymbol{\xi}_k(t) = [\boldsymbol{q}_k(t) + \boldsymbol{\xi}_k(t)] - \boldsymbol{q}_\mathrm{d}(t) \,. \quad (8.140)$$

Thus the learning control scheme may fail. In order to enhance the robustness of P-type learning control, a forgetting factor is introduced in the form

$$\boldsymbol{\tau}_{k+1}(t) = (1-\alpha)\boldsymbol{\tau}_k(t) + \alpha\boldsymbol{\tau}_0(t) - \boldsymbol{\Phi}[\Delta \boldsymbol{q}_k(t) + \boldsymbol{\xi}_k(t)] \,. \quad (8.141)$$

The original idea of using a forgetting factor in learning control originated with [8.104].

It has been rigorously proven that P-type learning control with a forgetting factor guarantees convergence to a neighborhood of the desired one of size $O(\alpha)$. Moreover, if the content of a long-term memory is refreshed after every k trials, where k is of $O(1/\alpha)$, then the trajectories converge to an ε-neighborhood of the desired control goal. The size of ε is dependent on the magnitude of the initial setting error, the nonrepeatable fluctuations of the dynamics, and the measurement noise. For a detailed stability investigation, please refer to [8.105, 106].

8.11.3 Summary

By applying learning control, the performance of repetitive tasks (such as painting or pick-and-place operation) is improved by utilizing data gathered in the previous cycles. In this section, two learning control schemes were introduced. First, pure P-type learning control and its robustness problem were described. Then P-type learning control with a forgetting factor was presented, enhancing the robustness of learning control.

Further Reading

Rigorous and refined exploration of learning control is first discussed independently in [8.2, 12].

Video-References

VIDEO 25 Gain change of the PID controller
available from http://handbookofrobotics.org/view-chapter/08/videodetails/25

VIDEO 757 Safe human-robot cooperation
available from http://handbookofrobotics.org/view-chapter/08/videodetails/757

VIDEO 758 Virtual whiskers – Highly responsive robot collision avoidance
available from http://handbookofrobotics.org/view-chapter/08/videodetails/758

VIDEO 759 JediBot – Experiments in human-robot sword-fighting
available from http://handbookofrobotics.org/view-chapter/08/videodetails/759

VIDEO 760 Different jerk limits of robot arm trajectories
available from http://handbookofrobotics.org/view-chapter/08/videodetails/760

VIDEO 761 Sensor-based online trajectory generation
available from http://handbookofrobotics.org/view-chapter/08/videodetails/761

References

8.1 C. Canudas de Wit, B. Siciliano, G. Bastin: *Theory of Robot Control* (Springer, London 1996)

8.2 J.J. Craig: Adaptive Control of Mechanical Manipulators, Ph.D. Thesis (UMI Dissertation Information Service, Ann Arbor 1986)

8.3 R.J. Schilling: *Fundametals of Robotics: Analysis and Control* (Prentice Hall, Englewood Cliffs 1989)

8.4 L. Sciavicco, B. Siciliano: *Modeling and Control of Robot Manipulator* (McGraw-Hill, New York 1996)

8.5 M.W. Spong, M. Vidyasagar: *Robot Dynamics and Control* (Wiley, New York 1989)

8.6 M.W. Spong, F.L. Lewis, C.T. Abdallah (Eds.): *Robot Control* (IEEE, New York 1989)

8.7 C.H. An, C.G. Atkeson, J.M. Hollerbach: *Model-Based Control of a Robot Manipulator* (MIT Press, Cambridge, 1988)

8.8 R.M. Murray, Z. Xi, S.S. Sastry: *A Mathematical Introduction to Robotic Manipulation* (CRC, Boca Raton 1994)

8.9 T. Yoshikawa: *Foundations of Robotics* (MIT Press, Cambridge 1990)

8.10 O. Khatib: A unified approach for motion and force control of robot manipulators: The operational space formulation, IEEE J. Robotics Autom. **3**(1), 43–53 (1987)

8.11 J.Y.S. Luh, M.W. Walker, R.P.C. Paul: Resolved-acceleration control of mechanical manipulator, IEEE Trans. Autom. Control **25**(3), 468–474 (1980)

8.12 S. Arimoto, F. Miyazaki: Stability and robustness of PID feedback control for robot manipulators of sensory capability. In: *Robotics Research*, ed.

Avidbots Ex1016
Page 236 of 2154

by M. Brady, R. Paul (MIT Press, Cambridge 1984) pp. 783–799
8.13 L.C. Fu: Robust adaptive decentralized control of robot manipulators, IEEE Trans. Autom. Control **37**(1), 106–110 (1992)
8.14 H. Seraji: Decentralized adaptive control of manipulators: Theory, simulation, and experimentation, IEEE Trans. Robotics Autom. **5**(2), 183–201 (1989)
8.15 J.G. Ziegler, N.B. Nichols: Optimum settings for automatic controllers, Trans. ASME **64**, 759–768 (1942)
8.16 Y. Choi, W.K. Chung: *PID Trajectory Tracking Control for Mechanical Systems*, Lecture Notes in Control and Information Sciences, Vol. 289 (Springer, New York 2004)
8.17 R. Kelly: PD control with desired gravity compensation of robot manipulators: A review, Int. J. Robotics Res. **16**(5), 660–672 (1997)
8.18 M. Takegaki, S. Arimoto: A new feedback method for dynamic control of manipulators, Trans. ASME J. Dyn. Syst. Meas, Control **103**, 119–125 (1981)
8.19 P. Tomei: Adaptive PD controller for robot manipulators, IEEE Trans. Robotics Autom. **7**(4), 565–570 (1991)
8.20 R. Ortega, A. Loria, R. Kelly: A semi-globally stable output feedback PI^2 D regulator for robot manipulators, IEEE Trans. Autom. Control **40**(8), 1432–1436 (1995)
8.21 D. Angeli: Input-to-State stability of PD-controlled robotic systems, Automatica **35**, 1285–1290 (1999)
8.22 J.A. Ramirez, I. Cervantes, R. Kelly: PID regulation of robot manipulators: Stability and performance, Syst. Control Lett. **41**, 73–83 (2000)
8.23 R. Kelly: Global positioning of robot manipulators via PD control plus a class of nonlinear integral actions, IEEE Trans. Autom. Control **43**(7), 934–937 (1998)
8.24 Z. Qu, J. Dorsey: Robust tracking control of robots by a linear feedback law, IEEE Trans. Autom. Control **36**(9), 1081–1084 (1991)
8.25 H. Berghuis, H. Nijmeijer: Robust control of robots via linear estimated state feedback, IEEE Trans. Autom. Control **39**(10), 2159–2162 (1994)
8.26 Y. Choi, W.K. Chung, I.H. Suh: Performance and $\mathcal{H}_\infty$ optimality of PID trajectory tracking controller for Lagrangian systems, IEEE Trans. Robotics Autom. **17**(6), 857–869 (2001)
8.27 K. Aström, T. Hagglund: *PID Controllers: Theory, Design, and Tuning* (Instrument Society of America, Research Triangle Park 1995)
8.28 C.C. Yu: *Autotuning of PID Controllers: Relay Feedback Approach* (Springer, London 1999)
8.29 F.L. Lewis, C.T. Abdallah, D.M. Dawson: *Control of Robot Manipulators* (Macmillan, New York 1993)
8.30 A. Isidori: *Nonlinear Control Systems: An Introduction*, Lecture Notes in Control and Information Sciences, Vol. 72 (Springer, New York 1985)
8.31 H. Berghuis, H. Nijmeijer: A passivity approach to controller–observer design for robots, IEEE Trans. Robotics Autom. **9**, 740–754 (1993)
8.32 J.J. Slotine, W. Li: On the adaptive control of robot manipulators, Int. J. Robotics Res. **6**(3), 49–59 (1987)
8.33 G. Liu, A.A. Goldenberg: Comparative study of robust saturation–based control of robot manipulators: analysis and experiments, Int. J. Robotics Res. **15**(5), 473–491 (1996)
8.34 D.M. Dawson, M. Grabbe, F.L. Lewis: Optimal control of a modified computed–torque controller for a robot manipulator, Int. J. Robotics Autom. **6**(3), 161–165 (1991)
8.35 D.M. Dawson, Z. Qu, J. Duffie: Robust tracking control for robot manipulators: Theory, simulation and implementation, Robotica **11**, 201–208 (1993)
8.36 A. Jaritz, M.W. Spong: An experimental comparison of robust control algorithms on a direct drive manipulator, IEEE Trans. Control Syst. Technol. **4**(6), 627–640 (1996)
8.37 A. Isidori: *Nonlinear Control Systems*, 3rd edn. (Springer, New York 1995)
8.38 J.J. Slotine, W. Li: *Applied Nonlinear Control* (Prentice Hall, Englewood Cliffs 1991)
8.39 W.J. Rugh: *Linear System Theory*, 2nd edn. (Prentice Hall, Upper Saddle River 1996)
8.40 M.W. Spong, M. Vidyasagar: Robust microprocessor control of robot manipulators, Automatica **23**(3), 373–379 (1987)
8.41 H.K. Khalil: *Nonlinear Systems*, 3rd edn. (Prentice Hall, Upper Saddle River 2002)
8.42 M. Vidysagar: *Nonlinear Systems Analysis*, 2nd edn. (Prentice Hall, Englewood Ciffs 1993)
8.43 J.T. Wen: A unified perspective on robot control: The energy Lyapunov function approach, Int. J. Adapt. Control Signal Process. **4**, 487–500 (1990)
8.44 R. Ortega, M.W. Spong: Adaptive motion control of rigid robots: A tutorial, Automatica **25**(6), 877–888 (1989)
8.45 N. Sadegh, R. Horowitz: Stability and robustness analysis of a class of adaptive contollers for robotic manipulators, Int. J. Robotics Res. **9**(3), 74–92 (1990)
8.46 S. Dubowsky, D.T. DesForges: The application of model-reference adaptive control to robotic manipulators, ASME J. Dyn. Syst. Meas. Control **37**(1), 106–110 (1992)
8.47 S.H. Hsu, L.C. Fu: A fully adaptive decentralized control of robot manipulators, Automatica **42**, 1761–1767 (2008)
8.48 A. Balestrino, G. de Maria, L. Sciavicco: An adaptive model following control for robotic manipulators, ASME J. Dyn. Syst. Meas. Control **105**, 143–151 (1983)
8.49 S. Nicosia, P. Tomei: Model reference adaptive control algorithms for industrial robots, Automatica **20**, 635–644 (1984)
8.50 R. Horowitz, M. Tomizuka: An adaptive control scheme for mechanical manipulators-Compensation of nonlinearity and decoupling control, ASME J. Dyn. Syst. Meas. Control **108**, 127–135 (1986)
8.51 I.D. Laudau: *Adaptive Control: The Model Reference Approach* (Dekker, New York 1979)
8.52 R. Lozano, C. Canudas de Wit: Passivity based adaptive control for mechanical manipulators using LS type estimation, IEEE Trans. Autom. Control **35**(12), 1363–1365 (1990)

Avidbots Ex1016
Page 237 of 2154

8.53 B. Brogliato, I.D. Laudau, R. Lozano: Passive least squares type estimation algorithm for direct adaptive control, Int. J. Adapt. Control Signal Process. **6**, 35–44 (1992)

8.54 R. Johansson: Adaptive control of robot manipulator motion, IEEE Trans. Robotics Autom. **6**(4), 483–490 (1990)

8.55 M.W. Walker: Adaptive control of manipulators containing closed kinematic loops, IEEE Trans. Robotics Autom. **6**(1), 10–19 (1990)

8.56 J.S. Reed, P.A. Ioannou: Instability analysis and robust adaptive control of robotic manipulators, IEEE Trans. Autom. Control **5**(3), 74–92 (1989)

8.57 G. Tao: On robust adaptive control of robot manipulators, Automatica **28**(4), 803–807 (1992)

8.58 H. Berghuis, R. Ogata, H. Nijmeijer: A robust adaptive controller for robot manipulators, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (1992) pp. 1876–1881

8.59 R. Johansson: Quadratic optimization of motion coordination and control, IEEE Trans. Autom. Control **35**(11), 1197–1208 (1990)

8.60 Z. Qu, D.M. Dawson: *Robust Tracking Control of Robot Manipulators* (IEEE, Piscataway 1996)

8.61 P. Dorato, C. Abdallah, V. Cerone: *Linear-Quadratic Control* (Prentice Hall, Upper Saddle River 1995)

8.62 A. Locatelli: *Optimal Control: An Introduction* (Birkhäuser, Basel 2001)

8.63 A. Isidori: Feedback control of nonlinear systems, Int. J. Robust Nonlin. Control **2**, 291–311 (1992)

8.64 A.J. der van Schaft: Nonlinear state space $\mathcal{H}_\infty$ control theory. In: *Essays on Control: Perspective in Theory and its Applications*, ed. by H.L. Trentelman, J.C. Willems (Birkhäuser, Basel 1993) pp. 153–190

8.65 A.J. der van Schaft: L_2-gain analysis of nonlinear systems and nonlinear state feedback $\mathcal{H}_\infty$ control, IEEE Trans. Autom. Control **37**(6), 770–784 (1992)

8.66 J. Park, W.K. Chung, Y. Youm: Analytic nonlinear $\mathcal{H}_\infty$ inverse-optimal control for Euler–Lagrange system, IEEE Trans. Robotics Autom. **16**(6), 847–854 (2000)

8.67 B.S. Chen, T.S. Lee, J.H. Feng: A nonlinear $\mathcal{H}_\infty$ control design in robotics systems under parametric perturbation and external disturbance, Int. J. Control **59**(12), 439–461 (1994)

8.68 J. Park, W.K. Chung: Design of a robust $\mathcal{H}_\infty$ PID control for industrial manipulators, ASME J. Dyn. Syst. Meas. Control **122**(4), 803–812 (2000)

8.69 R.H. Castain, R.P. Paul: An on-line dynamic trajectory generator, Int. J. Robotics Res. **3**(1), 68–72 (1984)

8.70 T. Kröger: *On-Line Trajectory Generation in Robotic Systems*, Springer Tracts in Advanced Robotics, Vol. 58 (Springer, Berlin, Heidelberg 2010)

8.71 D. Simon, C. Isik: A trigonometric trajectory generator for robotic arms, Int. J. Control **57**(3), 505–517 (1993)

8.72 L. Biagiotti, C. Melchiorri: *Trajectory Planning for Automatic Machines and Robots* (Springer, Berlin, Heidelberg 2008)

8.73 J.E. Bobrow: Optimal robot path planning using the minimum-time criterion, IEEE J. Robotics Autom. **4**(4), 443–450 (1988)

8.74 K.G. Shin, N.D. McKay: Minimum-time control of robotic manipulators with geometric path constraints, IEEE Trans. Autom. Control **30**(5), 531–541 (1985)

8.75 F. Pfeiffer, R. Johanni: A concept for manipulator trajectory planning, Proc. Int. IEEE Conf. Robotics Autom. (ICRA) (1986) pp. 1399–1405

8.76 W. Khalil, E. Dombre: Trajectory generation. In: *Modeling, Identification and Control of Robots*, ed. by W. Khalil, E. Dombre (Butterworth-Heinemann, Oxford 2004)

8.77 A.I. Kostrikin, Y.I. Manin: *Linear Algebra and Geometry* (Gordon and Breach Sci. Publ., Amsterdam 1997)

8.78 M.E. Kahn, B. Roth: The near-minimum-time control of open-loop articulated kinematic chains, ASME J. Dyn. Syst. Meas. Control **93**, 164–172 (1971)

8.79 M. Brady: Trajectory planning. In: *Robot Motion: Planning and Control*, ed. by M. Brady, J.M. Hollerbach, T.L. Johnson, T. Lozano-Pérez, M.T. Mason (MIT Press, Cambridge 1982)

8.80 R.P.C. Paul: Manipulator cartesian path control. In: *Robot Motion: Planning and Control*, ed. by M. Brady, J.M. Hollerbach, T.L. Johnson, T. Lozano-Pérez, M.T. Mason (MIT Press, Cambridge 1982)

8.81 R.H. Taylor: Planning and execution of straight-line manipulator trajectories. In: *Robot Motion: Planning and Control*, ed. by M. Brady, J.M. Hollerbach, T.L. Johnson, T. Lozano-Pérez, M.T. Mason (MIT Press, Cambridge 1982)

8.82 C.-S. Lin, P.-R. Chang, J.Y.S. Luh: Formulation and optimization of cubic polynomial joint trajectories for industrial robots, IEEE Trans. Autom. Control **28**(12), 1066–1074 (1983)

8.83 J.M. Hollerbach: Dynamic scaling of manipulator trajectories, ASME J. Dyn. Syst. Meas. Control **106**(1), 102–106 (1984)

8.84 K.J. Kyriakopoulos, G.N. Sridis: Minimum jerk path generation, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (1988) pp. 364–369

8.85 J.-J.E. Slotine, H.S. Yang: Improving the efficiency of time-optimal path-following algorithms, IEEE Trans. Robotics Autom. **5**(1), 118–124 (1989)

8.86 Z. Shiller, H.-H. Lu: Computation of path constrained time optimal motions with dynamic singularities, ASME J. Dyn. Syst. Meas. Control **114**(1), 34–40 (1992)

8.87 P. Fiorini, Z. Shiller: Time optimal trajectory planning in dynamic environments, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (1996) pp. 1553–1558

8.88 O. Dahl, L. Nielsen: Torque limited path following by on-line trajectory time scaling, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (1989) pp. 1122–1128

8.89 B. Cao, G.I. Dodds, G.W. Irwin: Time-optimal and smooth constrained path planning for robot manipulators, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (1994) pp. 1853–1858

8.90 B. Cao, G.I. Dodds, G.W. Irwin: A practical approach to near time-optimal inspection-task-sequence

Avidbots Ex1016
Page 238 of 2154

planning for two cooperative industrial robot arms, Int. J. Robotics Res. **17**(8), 858–867 (1998)
8.91 D. Constantinescu, E.A. Croft: Smooth and time-optimal trajectory planning for industrial manipulators along specified paths, J. Robotics Syst. **17**(5), 233–249 (2000)
8.92 S. Macfarlane, E.A. Croft: Jerk-bounded manipulator trajectory planning: Design for real-time applications, IEEE Trans. Robotics Autom. **19**(1), 42–52 (2003)
8.93 R.L. Andersson: *A Robot Ping-Pong Player: Experiment in Real-Time Intelligent Control* (MIT Press, Cambridge 1988)
8.94 R.L. Andersson: Aggressive trajectory generator for a robot ping-pong player, IEEE Control Syst. Mag. **9**(2), 15–21 (1989)
8.95 J. Lloyd, V. Hayward: Trajectory generation for sensor-driven and time-varying tasks, Int. J. Robotics Res. **12**(4), 380–393 (1993)
8.96 K. Ahn, W.K. Chung, Y. Yourn: Arbitrary states polynomial-like trajectory (ASPOT) generation, Proc. IEEE 30th Annu. Conf. Ind. Electron. Soc. (2004) pp. 123–128
8.97 X. Broquère, D. Sidobre, I. Herrera-Aguilar: Soft motion trajectory planner for service manipulator robot, Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS) (2008) pp. 2808–2813
8.98 R. Haschke, E. Weitnauer, H. Ritter: On-line planning of time-optimal, jerk-limited trajectories, Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS) (2008) pp. 3248–3253
8.99 J.J. Craig: *Introduction to Robotics: Mechanics and Control* (Prentice Hall, Upper Saddle River 2003)
8.100 K.S. Fu, R.C. Gonzalez, C.S.G. Lee: *Robotics: Control, Sensing, Vision and Intelligence* (McGraw-Hill, New York 1988)
8.101 M.W. Spong, S.A. Hutchinson, M. Vidyasagar: *Robot Modeling and Control* (Wiley, New York 2006)
8.102 S. Arimoto: Mathematical theory or learning with application to robot control. In: *Adaptive and Learning Control*, ed. by K.S. Narendra (Plenum, New York 1986) pp. 379–388
8.103 S. Kawamura, F. Miyazaki, S. Arimoto: Realization of robot motion based on a learning method, IEEE Trans. Syst. Man. Cybern. **18**(1), 126–134 (1988)
8.104 G. Heinzinger, D. Frewick, B. Paden, F. Miyazaki: Robust learning control, Proc. IEEE Int. Conf. Decis. Control (1989)
8.105 S. Arimoto: Robustness of learning control for robot manipulators, Proc. IEEE Int. Conf. Decis. Control (1990) pp. 1523–1528
8.106 S. Arimoto, T. Naiwa, H. Suzuki: Selective learning with a forgetting factor for robotic motion control, Proc. IEEE Int. Conf. Decis. Control (1991) pp. 728–733

Avidbots Ex1016
Page 239 of 2154

Multimedia Contents



9. Force Control

Luigi Villani, Joris De Schutter

A fundamental requirement for the success of a manipulation task is the capability to handle the physical contact between a robot and the environment. Pure motion control turns out to be inadequate because the unavoidable modeling errors and uncertainties may cause a rise of the contact force, ultimately leading to an unstable behavior during the interaction, especially in the presence of rigid environments. Force feedback and force control becomes mandatory to achieve a robust and versatile behavior of a robotic system in poorly structured environments as well as safe and dependable operation in the presence of humans. This chapter starts from the analysis of *indirect force control* strategies, conceived to keep the contact forces limited by ensuring a suitable compliant behavior to the end effector, without requiring an accurate model of the environment. Then the problem of *interaction tasks* modeling is analyzed, considering both the case of a rigid environment and the case of a compliant environment. For the specification of an interaction task, natural constraints set by the task geometry and artificial constraints set by the control strategy are established, with respect to suitable task frames. This formulation is the essential premise to the synthesis of *hybrid force/motion control* schemes.

9.1 **Background** 195
9.1.1 From Motion Control to Interaction Control 196
9.1.2 From Indirect Force Control to Hybrid Force/Motion Control 197

9.2 **Indirect Force Control** 198
9.2.1 Stiffness Control 198
9.2.2 Impedance Control 201

9.3 **Interaction Tasks** 205
9.3.1 Rigid Environment 205
9.3.2 Compliant Environment 208
9.3.3 Task Specification 209
9.3.4 Sensor-Based Contact Model Estimation 211

9.4 **Hybrid Force/Motion Control** 211
9.4.1 Acceleration-Resolved Approach 211
9.4.2 Passivity-Based Approach 214
9.4.3 Velocity-Resolved Approach 215

9.5 **Conclusions and Further Reading** 216
9.5.1 Indirect Force Control 216
9.5.2 Task Specification 217
9.5.3 Hybrid Force/Motion Control 217

Video-References 217

References 218

9.1 Background

Research on robot force control has flourished in the past three decades. Such a wide interest is motivated by the general desire of providing robotic systems with enhanced sensory capabilities. Robots using force, touch, distance, and visual feedback are expected to autonomously operate in unstructured environments other than the typical industrial shop floor.

Since the early work on telemanipulation, the use of force feedback was conceived to assist the human operator in the remote handling of objects with a slave manipulator. More recently, cooperative robot systems have been developed where two or more manipulators (viz. the fingers of a dexterous robot hand) are to be controlled so as to limit the exchanged forces

Avidbots Ex1016
Page 240 of 2154

and avoid squeezing of a commonly held object. Force control plays a fundamental role also in the achievement of robust and versatile behavior of robotic systems in open-ended environments, providing intelligent response in unforeseen situations and enhancing human–robot interaction.

9.1.1 From Motion Control to Interaction Control

Control of the physical interaction between a robot manipulator and the environment is crucial for the successful execution of a number of practical tasks where the robot end-effector has to manipulate an object or perform some operation on a surface. Typical examples in industrial settings include polishing, deburring, machining or assembly. A complete classification of possible robot tasks, considering also nonindustrial applications, is practically infeasible in view of the large variety of cases that may occur, nor would such a classification be really useful to find a general strategy to control the interaction with the environment.

During contact, the environment may set constraints on the geometric paths that can be followed by the end-effector, denoted as *kinematic constraints*. This situation, corresponding to the contact with a stiff surface, is generally referred to as *constrained motion*. In other cases, the contact task is characterized by a dynamic interaction between the robot and the environment that can be inertial (as in pushing a block), dissipative (as in sliding on a surface with friction) or elastic (as in pushing against an elastically compliant wall). In all these cases, the use of a pure motion control strategy for controlling interaction is prone to failure, as explained below.

Successful execution of an interaction task with the environment by using motion control could be obtained only if the task were accurately planned. This would in turn require an accurate model of both the robot manipulator (kinematics and dynamics) and the environment (geometry and mechanical features). A manipulator model may be known with sufficient precision, but a detailed description of the environment is difficult to obtain.

To understand the importance of task planning accuracy, it is sufficient to observe that in order to perform a mechanical part mating with a positional approach the relative positioning of the parts should be guaranteed with an accuracy of an order of magnitude greater than part mechanical tolerance. Once the absolute position of one part is exactly known, the manipulator should guide the motion of the other with the same accuracy.

In practice, the planning errors may give rise to a contact force and moment, causing a deviation of the end-effector from the desired trajectory. On the other hand, the control system reacts to reduce such deviations. This ultimately leads to a build-up of the contact force until saturation of the joint actuators is reached or breakage of the parts in contact occurs.

The higher the environment stiffness and position control accuracy are, the more easily a situation like the one just described can occur. This drawback can be overcome if a *compliant behavior* is ensured during the interaction. This compliant behavior can be achieved either in a passive or in an active fashion.

Passive Interaction Control

In passive interaction control the trajectory of the robot end-effector is modified by the interaction forces due to the inherent compliance of the robot. The compliance may be due to the structural compliance of the links, joints, and end-effector, or to the compliance of the position servo. Soft robot arms with elastic joints or links are purposely designed for intrinsically safe interaction with humans. In industrial applications, a mechanical device with passive compliance, known as the remote center of compliance (RCC) device [9.1], is widely adopted. An RCC is a compliant end-effector mounted on a rigid robot, designed and optimized for peg-into-hole assembly operations.

The passive approach to interaction control is very simple and cheap, because it does not require force/torque sensors; also, the preprogrammed trajectory of the end-effector must not be changed at execution time; moreover, the response of a passive compliance mechanism is much faster than active repositioning by a computer control algorithm. However, the use of passive compliance in industrial applications lacks flexibility, since for every robotic task a special-purpose compliant end-effector has to be designed and mounted. Also, it can only deal with small position and orientation deviations of the programmed trajectory. Finally, since no forces are measured, it can not guarantee that high contact forces will never occur.

Active Interaction Control

In active interaction control, the compliance of the robotic system is mainly ensured by a purposely designed control system. This approach usually requires the measurement of the contact force and moment, which are fed back to the controller and used to modify or even generate online the desired trajectory of the robot end-effector.

Active interaction control may overcome the aforementioned disadvantages of passive interaction control, but it is usually slower, more expensive, and more sophisticated. To obtain a reasonable task execution speed and disturbance rejection capability, active interaction

Avidbots Ex1016
Page 241 of 2154

control has to be used in combination with some degree of passive compliance [9.2]: feedback, by definition, always comes after a motion and force error has occurred, hence some passive compliance is needed in order to keep the reaction forces below an acceptable threshold.

Force Measurements

For a general force-controlled task, six force components are required to provide complete contact force information: three translational force components and three torques. Often, a force/torque sensor is mounted at the robot wrist [9.3], but other possibilities exist, for example, force sensors can be placed on the fingertips of robotic hands [9.4]; also, external forces and moments can be estimated via shaft torque measurements of joint torque sensors [9.5, 6]. However, the majority of the applications of force control (including industrial applications) is concerned with wrist force/torque sensors. In this case, the weight and inertia of the tool mounted between the sensor and the environment (i. e., the robot end-effector) is assumed to be negligible or suitably compensated from the force/torque measurements. The force signals may be obtained using strain measurements, which results in a stiff sensor, or deformation measurements (e.g., optically), resulting in a compliant sensor. The latter approach has an advantage if additional passive compliance is desired.

9.1.2 From Indirect Force Control to Hybrid Force/Motion Control

Active interaction control strategies can be grouped into two categories: those performing indirect force control and those performing direct force control. The main difference between the two categories is that the former achieve force control via motion control, without explicit closure of a force feedback loop; the latter instead offer the possibility of controlling the contact force and moment to a desired value, thanks to the closure of a force feedback loop.

To the first category belongs *impedance control* (or *admittance control*) [9.7, 8], where the deviation of the end-effector motion from the desired motion due to the interaction with the environment is related to the contact force through a mechanical impedance/admittance with adjustable parameters. A robot manipulator under impedance (or admittance) control is described by an equivalent mass–spring–damper system with adjustable parameters. This relationship is an impedance if the robot control reacts to the motion deviation by generating forces, while it corresponds to an admittance if the robot control reacts to interaction forces by imposing a deviation from the desired motion. Special cases of impedance and admittance control are *stiffness control* and *compliance control* [9.9], respectively, where only the static relationship between the end-effector position and orientation deviation from the desired motion and the contact force and moment is considered. Notice that, in the robot control literature, the terms impedance control and admittance control are often used to refer to the same control scheme; the same happens for stiffness and compliance control. Moreover, if only the relationship between the contact force and moment and the end-effector linear and angular velocity is of interest, the corresponding control scheme is referred to as *damping control* [9.10].

Indirect force control schemes do not require, in principle, measurements of contact forces and moments; the resulting impedance or admittance is typically nonlinear and coupled. However, if a force/torque sensor is available, then force measurements can be used in the control scheme to achieve a linear and decoupled behavior.

Differently from indirect force control, direct force control requires an explicit model of the interaction task. In fact, the user has to specify the desired motion and the desired contact force and moment in a consistent way with respect to the constraints imposed by the environment. A widely adopted strategy belonging to this category is *hybrid force/motion control*, which aims at controlling the motion along the unconstrained task directions and force (and moment) along the constrained task directions. The starting point is the observation that, for many robotic tasks, it is possible to introduce an orthogonal reference frame, known as the compliance frame [9.11] (or task frame [9.12]) which allows one to specify the task in terms of natural and artificial constrains acting along and about the three orthogonal axes of this frame. Based on this decomposition, hybrid force/motion control allows simultaneous control of both the contact force and the end-effector motion in two mutually independent subspaces. Simple selection matrices acting on both the desired and feedback quantities serve this purpose for planar contact surfaces [9.13], whereas suitable projection matrices must be used for general contact tasks, which can also be derived from the explicit constraint equations [9.14–16]. Several implementation of hybrid motion control schemes are available, e.g., based on inverse dynamics control in the operational space [9.17], passivity-based control [9.18], or outer force control loops closed around inner motion loops, typically available in industrial robots [9.2].

If an accurate model of the environment is not available, the force control action and the motion control action can be superimposed, resulting in a *parallel force/position control* scheme. In this approach, the

Avidbots Ex1016
Page 242 of 2154

force controller is designed so as to dominate the motion controller; hence, a position error would be tolerated along the constrained task directions in order to ensure force regulation [9.19].

9.2 Indirect Force Control

To gain insight into the problems arising at the interaction between the end-effector of a robot manipulator and the environment, it is worth analyzing the effects of a motion control strategy in the presence of a contact force and moment. To this aim, assume that a reference frame Σ_e is attached to the end-effector, and let $\boldsymbol{p}_e$ denote the position vector of the origin and $\mathbf{R}_e$ the rotation matrix with respect to a fixed base frame. The end-effector velocity is denoted by the 6×1 twist vector $\boldsymbol{v}_e = (\dot{\boldsymbol{p}}_e^T \ \boldsymbol{\omega}_e^T)^T$ where $\dot{\boldsymbol{p}}_e$ is the translational velocity and $\boldsymbol{\omega}_e$ the angular velocity, and can be computed from the $n\times 1$ joint velocity vector $\dot{\boldsymbol{q}}$ using the linear mapping

$$\boldsymbol{v}_e = \mathbf{J}(\boldsymbol{q})\dot{\boldsymbol{q}} \,. \tag{9.1}$$

The matrix $\mathbf{J}$ is the $6\times n$ end-effector geometric Jacobian. For simplicity, the case of nonredundant nonsingular manipulators is considered; therefore, $n = 6$ and the Jacobian is a square nonsingular matrix. The force $\boldsymbol{f}_e$ and moment $\boldsymbol{m}_e$ applied by the end-effector to the environment are the components of the wrench $\boldsymbol{h}_e = (\boldsymbol{f}_e^T \ \boldsymbol{m}_e^T)^T$.

It is useful to consider the operational space formulation of the dynamic model of a rigid robot manipulator in contact with the environment

$$\boldsymbol{\Lambda}(\boldsymbol{q})\dot{\boldsymbol{v}}_e + \boldsymbol{\Gamma}(\boldsymbol{q},\dot{\boldsymbol{q}})\boldsymbol{v}_e + \boldsymbol{\eta}(\boldsymbol{q}) = \boldsymbol{h}_c - \boldsymbol{h}_e \,, \tag{9.2}$$

where

$$\boldsymbol{\Lambda}(\boldsymbol{q}) = (\mathbf{J}\mathbf{H}(\boldsymbol{q})^{-1}\mathbf{J}^T)^{-1}$$

is the 6×6 operational space inertia matrix,

$$\boldsymbol{\Gamma}(\boldsymbol{q},\dot{\boldsymbol{q}}) = \mathbf{J}^{-T}\mathbf{C}(\boldsymbol{q},\dot{\boldsymbol{q}})\mathbf{J}^{-1} - \boldsymbol{\Lambda}(\boldsymbol{q})\dot{\mathbf{J}}\mathbf{J}^{-1}$$

is the wrench including centrifugal and Coriolis effects, and $\boldsymbol{\eta}(\boldsymbol{q}) = \mathbf{J}^{-T}\boldsymbol{g}(\boldsymbol{q})$ is the wrench of the gravitational effects; $\mathbf{H}(\boldsymbol{q})$, $\mathbf{C}(\boldsymbol{q},\dot{\boldsymbol{q}})$ and $\boldsymbol{g}(\boldsymbol{q})$ are the corresponding quantities defined in the joint space. The vector $\boldsymbol{h}_c = \mathbf{J}^{-T}\boldsymbol{\tau}$ is the equivalent end-effector wrench corresponding to the input joint torques $\boldsymbol{\tau}$.

9.2.1 Stiffness Control

In the classical operational space formulation, the end-effector position and orientation is described by a 6×1 vector $\boldsymbol{x}_e = (\boldsymbol{p}_e^T \ \boldsymbol{\varphi}_e^T)^T$, where $\boldsymbol{\varphi}_e$ is a set of Euler angles extracted from $\mathbf{R}_e$. Hence, a desired end-effector position and orientation can be assigned in terms of a vector $\boldsymbol{x}_d$, corresponding to the position of the origin $\boldsymbol{p}_d$ and the rotation matrix $\mathbf{R}_d$ of a desired frame Σ_d. The end-effector error can be denoted as $\Delta\boldsymbol{x}_{de} = \boldsymbol{x}_d - \boldsymbol{x}_e$, and the corresponding velocity error, assuming a constant $\boldsymbol{x}_d$, can be expressed as $\Delta\dot{\boldsymbol{x}}_{de} = -\dot{\boldsymbol{x}}_e = -\mathbf{A}^{-1}(\boldsymbol{\varphi}_e)\boldsymbol{v}_e$, with

$$\mathbf{A}(\boldsymbol{\varphi}_e) = \begin{pmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{T}(\boldsymbol{\varphi}_e) \end{pmatrix} ,$$

where $\mathbf{I}$ is the 3×3 identity matrix, $\mathbf{0}$ is a 3×3 null matrix, and $\mathbf{T}$ is the 3×3 matrix of the mapping $\boldsymbol{\omega}_e = \mathbf{T}(\boldsymbol{\varphi}_e)\dot{\boldsymbol{\varphi}}_e$, depending on the particular choice of the Euler angles.

Consider the motion control law

$$\boldsymbol{h}_c = \mathbf{A}^{-T}(\boldsymbol{\varphi}_e)\mathbf{K}_P\Delta\boldsymbol{x}_{de} - \mathbf{K}_D\boldsymbol{v}_e + \boldsymbol{\eta}(\boldsymbol{q}) \,, \tag{9.3}$$

corresponding to a simple proportional–derivative (PD) + gravity compensation control in the operational space, where $\mathbf{K}_P$ and $\mathbf{K}_D$ are symmetric and positive-definite 6×6 matrices.

In the absence of interaction with the environment (i. e., when $\boldsymbol{h}_e = \mathbf{0}$), the equilibrium $\boldsymbol{v}_e = \mathbf{0}$, $\Delta\boldsymbol{x}_{de} = \mathbf{0}$ for the closed-loop system, corresponding to the desired position and orientation for the end-effector, is asymptotically stable. The stability proof is based on the positive-definite Lyapunov function

$$V = \frac{1}{2}\boldsymbol{v}_e^T\boldsymbol{\Lambda}(\boldsymbol{q})\boldsymbol{v}_e + \frac{1}{2}\Delta\boldsymbol{x}_{de}\mathbf{K}_P\Delta\boldsymbol{x}_{de} \,,$$

whose time derivative along the trajectories of the closed-loop system is the negative semidefinite function

$$\dot{V} = -\boldsymbol{v}_e^T\mathbf{K}_D\boldsymbol{v}_e \,. \tag{9.4}$$

In the presence of a constant wrench $\boldsymbol{h}_e$, using a similar Lyapunov argument, a different asymptotically stable equilibrium can be found, with a nonnull $\Delta\boldsymbol{x}_{de}$. The new equilibrium is the solution of the equation

$$\mathbf{A}^{-T}(\boldsymbol{\varphi}_e)\mathbf{K}_P\Delta\boldsymbol{x}_{de} - \boldsymbol{h}_e = \mathbf{0} \,,$$

Avidbots Ex1016
Page 243 of 2154

which can be written in the form

$$\Delta \boldsymbol{x}_{\mathrm{de}} = \mathbf{K}_{\mathrm{P}}^{-1}\mathbf{A}^{\mathrm{T}}(\boldsymbol{\varphi}_{\mathrm{e}})\boldsymbol{h}_{\mathrm{e}} \,, \tag{9.5}$$

or, equivalently, as

$$\boldsymbol{h}_{\mathrm{e}} = \mathbf{A}^{-\mathrm{T}}(\boldsymbol{\varphi}_{\mathrm{e}})\mathbf{K}_{\mathrm{P}}\Delta \boldsymbol{x}_{\mathrm{de}} \,. \tag{9.6}$$

Equation (9.6) shows that in the steady state the end-effector, under a proportional control action on the position and orientation error, behaves as a six-degree-of-freedom (DOF) spring in respect of the external force and moment $\boldsymbol{h}_{\mathrm{e}}$. Thus, the matrix $\mathbf{K}_{\mathrm{P}}$ plays the role of an *active stiffness*, meaning that it is possible to act on the elements of $\mathbf{K}_{\mathrm{P}}$ so as to ensure a suitable elastic behavior of the end-effector during the interaction. Analogously, (9.5) represents a compliance relationship, where the matrix $\mathbf{K}_{\mathrm{P}}^{-1}$ plays the role of an *active compliance*. This approach, consisting of assigning a desired position and orientation and a suitable static relationship between the deviation of the end-effector position and orientation from the desired motion and the force exerted on the environment, is known as *stiffness control*.

The selection of the stiffness/compliance parameters is not easy, and strongly depends on the task to be executed. A higher value of the active stiffness means a higher accuracy of the position control at the expense of higher interaction forces. Hence, if it is expected to meet some physical constraint in a particular direction, the end-effector stiffness in that direction should be made low to ensure low interaction forces. Conversely, along the directions where physical constraints are not expected, the end-effector stiffness should be made high so as to follow closely the desired position. This allows discrepancies between the desired and achievable positions due to the constraints imposed by the environment to be resolved without excessive contact forces and moments.

It must be pointed out, however, that a selective stiffness behavior along different directions cannot be effectively assigned in practice on the basis of (9.6). This can easily be understood by using the classical definition of a mechanical stiffness for two bodies connected by a 6-DOF spring, in terms of the linear mapping between the infinitesimal twist displacement of the two bodies at an unloaded equilibrium and the elastic wrench.

In the case of the active stiffness, the two bodies are, respectively, the end-effector, with the attached frame Σ_{e}, and a virtual body, attached to the desired frame Σ_{d}. Hence, from (9.6), the following mapping can be derived

$$\boldsymbol{h}_{\mathrm{e}} = \mathbf{A}^{-\mathrm{T}}(\boldsymbol{\varphi}_{\mathrm{e}})\mathbf{K}_{\mathrm{P}}\mathbf{A}^{-1}(\boldsymbol{\varphi}_{\mathrm{e}})\delta \boldsymbol{x}_{\mathrm{de}} \,, \tag{9.7}$$

in the case of an infinitesimal twist displacement $\delta \boldsymbol{x}_{\mathrm{de}}$ defined as

$$\delta \boldsymbol{x}_{\mathrm{de}} = \begin{pmatrix} \delta \boldsymbol{p}_{\mathrm{de}} \\ \delta \boldsymbol{\theta}_{\mathrm{de}} \end{pmatrix} = \begin{pmatrix} \Delta \dot{\boldsymbol{p}}_{\mathrm{de}} \\ \Delta \boldsymbol{\omega}_{\mathrm{de}} \end{pmatrix} \mathrm{d}t = -\begin{pmatrix} \dot{\boldsymbol{p}}_{\mathrm{e}} \\ \boldsymbol{\omega}_{\mathrm{e}} \end{pmatrix} \mathrm{d}t \,,$$

where $\Delta \dot{\boldsymbol{p}}_{\mathrm{de}} = \dot{\boldsymbol{p}}_{\mathrm{d}} - \dot{\boldsymbol{p}}_{\mathrm{e}}$ is the time derivative of the position error $\Delta \boldsymbol{p}_{\mathrm{de}} = \boldsymbol{p}_{\mathrm{d}} - \boldsymbol{p}_{\mathrm{e}}$ and $\Delta \boldsymbol{\omega}_{\mathrm{de}} = \boldsymbol{\omega}_{\mathrm{d}} - \boldsymbol{\omega}_{\mathrm{e}}$ is the angular velocity error. Equation (9.7) shows that the actual stiffness matrix is $\mathbf{A}^{-\mathrm{T}}(\boldsymbol{\varphi}_{\mathrm{e}})\mathbf{K}_{\mathrm{P}}\mathbf{A}^{-1}(\boldsymbol{\varphi}_{\mathrm{e}})$, which depends on the end-effector orientation through the vector $\boldsymbol{\varphi}_{\mathrm{e}}$, so that, in practice, the selection of the stiffness parameters is quite difficult.

This problem can be overcome by defining a *geometrically consistent active stiffness*, with the same structure and properties as ideal mechanical springs.

Mechanical Springs

Consider two elastically coupled rigid bodies A and B and two reference frames Σ_{a} and Σ_{b}, attached to A and B, respectively. Assuming that at equilibrium frames Σ_{a} and Σ_{b} coincide, the compliant behavior near the equilibrium can be described by the linear mapping

$$\boldsymbol{h}_{\mathrm{b}}^{\mathrm{b}} = \mathbf{K}\delta \boldsymbol{x}_{\mathrm{ab}}^{\mathrm{b}} = \begin{pmatrix} \mathbf{K}_{\mathrm{t}} & \mathbf{K}_{\mathrm{c}} \\ \mathbf{K}_{\mathrm{c}}^{\mathrm{T}} & \mathbf{K}_{\mathrm{o}} \end{pmatrix} \delta \boldsymbol{x}_{\mathrm{ab}}^{\mathrm{b}} \,, \tag{9.8}$$

where $\boldsymbol{h}_{\mathrm{b}}^{\mathrm{b}}$ is the elastic wrench applied to body B, expressed in frame B, in the presence of an infinitesimal twist displacement $\delta \boldsymbol{x}_{\mathrm{ab}}^{\mathrm{b}}$ of frame Σ_{a} with respect to frame Σ_{b}, expressed in frame B. The elastic wrench and the infinitesimal twist displacement in (9.8) can also be expressed equivalently in frame Σ_{a}, since Σ_{a} and Σ_{b} coincide at equilibrium. Therefore, $\boldsymbol{h}_{\mathrm{b}}^{\mathrm{b}} = \boldsymbol{h}_{\mathrm{b}}^{\mathrm{a}}$ and $\delta \boldsymbol{x}_{\mathrm{ab}}^{\mathrm{b}} = \delta \boldsymbol{x}_{\mathrm{ab}}^{\mathrm{a}}$; moreover, for the elastic wrench applied to body A, $\boldsymbol{h}_{\mathrm{a}}^{\mathrm{a}} = \mathbf{K}_{\mathrm{t}}\delta \boldsymbol{x}_{\mathrm{ba}}^{\mathrm{a}} = -\boldsymbol{h}_{\mathrm{b}}^{\mathrm{b}}$ being $\delta \boldsymbol{x}_{\mathrm{ba}}^{\mathrm{a}} = -\delta \boldsymbol{x}_{\mathrm{ab}}^{\mathrm{b}}$. This property of the mapping (9.8) is known as *port symmetry*.

In (9.8), $\mathbf{K}$ is the 6×6 symmetric positive-semidefinite stiffness matrix. The 3×3 matrices $\mathbf{K}_{\mathrm{t}}$ and $\mathbf{K}_{\mathrm{o}}$, called respectively the *translational stiffness* and *rotational stiffness*, are also symmetric. It can be shown that, if the 3×3 matrix $\mathbf{K}_{\mathrm{c}}$, called the *coupling stiffness* is symmetric, there is maximum decoupling between rotation and translation. In this case, the point corresponding to the coinciding origins of the frames Σ_{a} and Σ_{b} is called the *center of stiffness*. Similar definitions and results can be formulated for the case of the compliance matrix $\mathbf{C} = \mathbf{K}^{-1}$. In particular, it is possible to define a *center of compliance* in the case that the off-diagonal blocks of the compliance matrix are symmetric. The center of stiffness and compliance do not necessarily coincide.

Avidbots Ex1016
Page 244 of 2154

There are special cases in which no coupling exists between translation and rotation, i. e., a relative translation of the bodies results in a wrench corresponding to a pure force along an axis through the center of stiffness; also, a relative rotation of the bodies results in a wrench that is equivalent to a pure torque about an axis through the centers of stiffness. In these cases, the center of stiffness and compliance coincide. Mechanical systems with completely decoupled behavior are, e.g., the remote center of compliance (RCC) devices.

Since $\mathbf{K}_t$ is symmetric, there exists a rotation matrix $\mathbf{R}_t$ with respect to the frame $\Sigma_a = \Sigma_b$ at equilibrium, such that $\mathbf{K}_t = \mathbf{R}_t\,\mathbf{\Gamma}_t\,\mathbf{R}_t^T$, and $\mathbf{\Gamma}_t$ is a diagonal matrix whose diagonal elements are the *principal translational stiffnessess* in the directions corresponding to the columns of the rotation matrix $\mathbf{R}_t$, known as the *principal axes of translational stiffness*. Analogously, $\mathbf{K}_o$ can be expressed as $\mathbf{K}_o = \mathbf{R}_o\mathbf{\Gamma}_o\mathbf{R}_o^T$, where the diagonal elements of $\mathbf{\Gamma}_o$ are the *principal rotational stiffnesses* about the axes corresponding to the columns of rotation matrix $\mathbf{R}_o$, known as the *principal axes of rotational stiffness*. Moreover, assuming that the origins of Σ_a and Σ_b at equilibrium coincide with the center of stiffness, the expression $\mathbf{K}_c = \mathbf{R}_c\mathbf{\Gamma}_c\mathbf{R}_c^T$ can be found, where the diagonal elements of $\mathbf{\Gamma}_c$ are the *principal coupling stiffnesses* along the directions corresponding to the columns of the rotation matrix $\mathbf{R}_c$, known as the *principal axes of coupling stiffness*. In sum, a 6×6 stiffness matrix can be specified, with respect to a frame with origin in the center of stiffness, in terms of the principal stiffness parameters and principal axes.

Notice that the mechanical stiffness defined by (9.8) describes the behavior of an ideal 6-DOF spring which stores potential energy. The potential energy function of an ideal stiffness depends only on the relative position and orientation of the two attached bodies and is port symmetric. A physical 6-DOF spring has a predominant behavior similar to the ideal one, but nevertheless it always has parasitic effects causing energy dissipation.

Geometrically Consistent Active Stiffness

To achieve a geometrically consistent 6-DOF active stiffness, a suitable definition of the proportional control action in control law (9.3) is required. This control action can be interpreted as the elastic wrench applied to the end-effector, in the presence of a *finite displacement* of the desired frame Σ_d with respect to the end-effector frame Σ_e. Hence, the properties of the ideal mechanical stiffness for small displacements should be extended to the case of finite displacements. Moreover, to guarantee asymptotic stability in the sense of Lyapunov, a suitable potential elastic energy function must be defined.

For simplicity, it is assumed that the coupling stiffness matrix is zero. Hence, the potential elastic energy can be computed as the sum of a translational potential energy and a rotational potential energy.

The translational potential energy can be defined as

$$V_t = \frac{1}{2}\Delta\boldsymbol{p}_{de}^T\mathbf{K}'_{Pt}\Delta\boldsymbol{p}_{de}\,, \tag{9.9}$$

with

$$\mathbf{K}'_{Pt} = \frac{1}{2}\mathbf{R}_d\mathbf{K}_{Pt}\mathbf{R}_d^T + \frac{1}{2}\mathbf{R}_e\mathbf{K}_{Pt}\mathbf{R}_e^T\,,$$

where $\mathbf{K}_{Pt}$ is a 3×3 symmetric positive-definite matrix. The use of $\mathbf{K}'_{Pt}$ in lieu of $\mathbf{K}_{Pt}$ in (9.9) guarantees that the potential energy is port symmetric also in the case of finite displacements. Matrices $\mathbf{K}'_{Pt}$ and $\mathbf{K}_{Pt}$ coincide at equilibrium (i. e., when $\mathbf{R}_d = \mathbf{R}_e$) and in the case of isotropic translational stiffness (i. e., when $\mathbf{K}_{Pt} = k_{Pt}\mathbf{I}$).

The computation of the power $\dot{V}_t$ yields

$$\dot{V}_t = \Delta\dot{\boldsymbol{p}}_{de}^{eT}\boldsymbol{f}_{\Delta t}^e + \Delta\boldsymbol{\omega}_{de}^{eT}\boldsymbol{m}_{\Delta t}^e\,,$$

where $\Delta\dot{\boldsymbol{p}}_{de}^e$ is the time derivative of the position displacement $\Delta\boldsymbol{p}_{de}^e = \mathbf{R}_e^T(\boldsymbol{p}_d - \boldsymbol{p}_e)$, while $\Delta\boldsymbol{\omega}_{de}^e = \mathbf{R}_e^T(\boldsymbol{\omega}_d - \boldsymbol{\omega}_e)$. The vectors $\boldsymbol{f}_{\Delta t}^e$ and $\boldsymbol{\mu}_{\Delta t}^e$ are, respectively, the elastic force and moment applied to the end-effector in the presence of the finite position displacement $\Delta\boldsymbol{p}_{de}^e$. These vectors have the following expressions when computed in the base frame

$$\boldsymbol{f}_{\Delta t} = \mathbf{K}'_{Pt}\Delta\boldsymbol{p}_{de} \qquad \boldsymbol{m}_{\Delta t} = \mathbf{K}''_{Pt}\Delta\boldsymbol{p}_{de}\,, \tag{9.10}$$

with

$$\mathbf{K}''_{Pt} = \frac{1}{2}\mathbf{S}(\Delta\boldsymbol{p}_{de})\mathbf{R}_d\mathbf{K}_{Pt}\mathbf{R}_d^T\,,$$

where $\mathbf{S}(\cdot)$ is the skew-symmetric operator performing the vector product. The vector $\boldsymbol{h}_{\Delta t} = (\boldsymbol{f}_{\Delta t}^T\ \boldsymbol{m}_{\Delta t}^T)^T$ is the elastic wrench applied to the end-effector in the presence of a finite position displacement $\Delta\boldsymbol{p}_{de}$ and a null orientation displacement. The moment $\boldsymbol{m}_{\Delta t}$ is null in the case of isotropic translational stiffness.

To define the rotational potential energy, a suitable definition of the orientation displacement between the frames Σ_d and Σ_e has to be adopted. A possible choice is the vector part of the unit quaternion $\{\eta_{de}, \boldsymbol{\epsilon}_{de}^e\}$ that can be extracted from matrix $\mathbf{R}_d^e = \mathbf{R}_e^T\mathbf{R}_d$. Hence, the orientation potential energy has the form

$$V_o = 2\boldsymbol{\epsilon}_{de}^{eT}\mathbf{K}_{Po}\boldsymbol{\epsilon}_{de}^e\,, \tag{9.11}$$

Avidbots Ex1016
Page 245 of 2154

where $\mathbf{K}_{\mathrm{Po}}$ is a 3×3 symmetric positive-definite matrix. The function V_{o} is port symmetric because $\boldsymbol{\epsilon}^{\mathrm{e}}_{\mathrm{de}} = -\boldsymbol{\epsilon}^{\mathrm{d}}_{\mathrm{ed}}$.

The computation of the power $\dot{V}_{\mathrm{o}}$ yields

$$\dot{V}_{\mathrm{o}} = \Delta\boldsymbol{\omega}^{\mathrm{eT}}_{\mathrm{de}}\boldsymbol{m}^{\mathrm{e}}_{\Delta\mathrm{o}} ,$$

where

$$\boldsymbol{m}_{\Delta\mathrm{o}} = \mathbf{K}'_{\mathrm{Po}}\boldsymbol{\epsilon}_{\mathrm{de}} , \tag{9.12}$$

with

$$\mathbf{K}'_{\mathrm{Po}} = 2\mathbf{E}^{\mathrm{T}}(\boldsymbol{\eta}_{\mathrm{de}}, \boldsymbol{\epsilon}_{\mathrm{de}})\mathbf{R}_{\mathrm{e}}\mathbf{K}_{\mathrm{Po}}\mathbf{R}^{\mathrm{T}}_{\mathrm{e}} ,$$

and $\mathbf{E}(\boldsymbol{\eta}_{\mathrm{de}}, \boldsymbol{\epsilon}_{\mathrm{de}}) = \boldsymbol{\eta}_{\mathrm{de}}\mathbf{I} - \mathbf{S}(\boldsymbol{\epsilon}_{\mathrm{de}})$. The above equations show that a finite orientation displacement $\boldsymbol{\epsilon}_{\mathrm{de}} = \mathbf{R}^{\mathrm{T}}_{\mathrm{e}}\boldsymbol{\epsilon}^{\mathrm{e}}_{\mathrm{de}}$ produces an elastic wrench $\boldsymbol{h}_{\Delta\mathrm{o}} = (\mathbf{0}^{\mathrm{T}} \;\; \boldsymbol{m}^{\mathrm{T}}_{\Delta\mathrm{o}})^{\mathrm{T}}$ which is equivalent to a pure moment.

Hence, the total elastic wrench in the presence of a finite position and orientation displacement of the desired frame Σ_{d} with respect to the end-effector frame Σ_{e} can be defined in the base frame as

$$\boldsymbol{h}_{\Delta} = \boldsymbol{h}_{\Delta\mathrm{t}} + \boldsymbol{h}_{\Delta\mathrm{o}} . \tag{9.13}$$

where $\boldsymbol{h}_{\Delta\mathrm{t}}$ and $\boldsymbol{h}_{\Delta\mathrm{o}}$ are computed according to (9.10) and (9.12), respectively.

Using (9.13) for the computation of the elastic wrench in the case of an infinitesimal twist displacement $\delta\boldsymbol{x}^{\mathrm{e}}_{\mathrm{de}}$ near the equilibrium, and discarding the high-order infinitesimal terms, yields the linear mapping

$$\boldsymbol{h}^{\mathrm{e}}_{\mathrm{e}} = \mathbf{K}_{\mathrm{P}}\delta\boldsymbol{x}^{\mathrm{e}}_{\mathrm{de}} = \begin{pmatrix} \mathbf{K}_{\mathrm{Pt}} & \mathbf{0} \\ \mathbf{0} & \mathbf{K}_{\mathrm{Po}} \end{pmatrix} \delta\boldsymbol{x}^{\mathrm{e}}_{\mathrm{de}} . \tag{9.14}$$

Therefore, $\mathbf{K}_{\mathrm{P}}$ represents the stiffness matrix of an ideal spring with respect to a frame Σ_{e} (coinciding with Σ_{d} at equilibrium) with the origin at the center of stiffness. Moreover, it can be shown, using definition (9.13), that the physical/geometrical meaning of the principal stiffnesses and of the principal axes for the matrices $\mathbf{K}_{\mathrm{Pt}}$ and $\mathbf{K}_{\mathrm{Po}}$ are preserved also in the case of large displacements.

The above results imply that the active stiffness matrix $\mathbf{K}_{\mathrm{P}}$ can be set in a geometrically consistent way with respect to the task at hand.

Notice that geometrical consistency can also be ensured with different definitions of the orientation error in the potential orientation energy (9.11), for example, any error based on the angle/axis representation of $\mathbf{R}^{\mathrm{d}}_{\mathrm{e}}$ can be adopted (the unit quaternion belongs to this category), or, more generally, homogeneous matrices or exponential coordinates (for the case of both position and orientation errors). Also, the *XYZ* Euler angles extracted from the matrix $\mathbf{R}^{\mathrm{d}}_{\mathrm{e}}$ could be used; however, in this case, it can be shown that the principal axes of rotational stiffness cannot be set arbitrarily but must coincide with those of the end-effector frame.

Stiffness control with a geometrically consistent active stiffness can be defined using the control law

$$\boldsymbol{h}_{\mathrm{c}} = \boldsymbol{h}_{\Delta} - \mathbf{K}_{\mathrm{D}}\boldsymbol{v}_{\mathrm{e}} + \boldsymbol{\eta}(\boldsymbol{q}) , \tag{9.15}$$

with $\boldsymbol{h}_{\Delta}$ in (9.13). The asymptotic stability about the equilibrium in the case $\boldsymbol{h}_{\mathrm{e}} = \mathbf{0}$ can be proven using the Lyapunov function

$$V = \frac{1}{2}\boldsymbol{v}^{\mathrm{T}}_{\mathrm{e}}\boldsymbol{\Lambda}(\boldsymbol{q})\boldsymbol{v}_{\mathrm{e}} + V_{\mathrm{t}} + V_{\mathrm{o}} ,$$

with V_{t} and V_{o} given in (9.9) and (9.11), respectively, whose time derivative along the trajectories of the closed-loop system, in case the frame Σ_{d} is motionless, has the same expression as in (9.4). When $\boldsymbol{h}_{\mathrm{e}} \neq \mathbf{0}$, a different asymptotically stable equilibrium can be found, corresponding to a nonnull displacement of the desired frame Σ_{d} with respect to the end-effector frame Σ_{e}. The new equilibrium is the solution of the equation $\boldsymbol{h}_{\Delta} = \boldsymbol{h}_{\mathrm{e}}$.

Stiffness control allows to keep the interaction force and moment limited at the expense of the end-effector position and orientation error, with a proper choice of the stiffness matrix, without the need of a force/torque sensor. However, in the presence of disturbances (e.g., joint friction) which can be modeled as an equivalent end-effector wrench, the adoption of low values for the active stiffness may produce large deviations with respect to the desired end-effector position and orientation, also in the absence of interaction with the environment.

9.2.2 Impedance Control

Stiffness control is designed to achieve a desired *static* behavior of the interaction. In fact, the dynamics of the controlled system depends on that of the robot manipulator, which is nonlinear and coupled. A more demanding objective may be that of achieving a desired *dynamic* behavior for the end-effector, e.g., that of a second-order mechanical system with six degrees of freedom, characterized by a given mass, damping, and stiffness, known as mechanical *impedance*.

The starting point to pursue this goal may be the acceleration-resolved approach used for motion control, which is aimed at decoupling and linearizing the nonlinear robot dynamics at the acceleration level via an inverse dynamics control law. In the presence of interaction with the environment, the control law

$$\boldsymbol{h}_{\mathrm{c}} = \boldsymbol{\Lambda}(\boldsymbol{q})\boldsymbol{\alpha} + \boldsymbol{\Gamma}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{\eta}(\boldsymbol{q}) + \boldsymbol{h}_{\mathrm{e}} \tag{9.16}$$

Avidbots Ex1016
Page 246 of 2154

cast into the dynamic model (9.2) results in

$$\dot{\boldsymbol{v}}_e = \boldsymbol{\alpha} \,, \tag{9.17}$$

where $\boldsymbol{\alpha}$ is a properly designed control input with the meaning of an acceleration referred to the base frame. Considering the identity $\dot{\boldsymbol{v}}_e = \bar{\mathbf{R}}_e^T \dot{\boldsymbol{v}}_e^e + \dot{\bar{\mathbf{R}}}_e^T \boldsymbol{v}_e^e$, with

$$\bar{\mathbf{R}}_e = \begin{pmatrix} \mathbf{R}_e & \mathbf{0} \\ \mathbf{0} & \mathbf{R}_e \end{pmatrix} \,,$$

the choice

$$\boldsymbol{\alpha} = \bar{\mathbf{R}}_e^T \boldsymbol{\alpha}^e + \dot{\bar{\mathbf{R}}}_e^T \boldsymbol{v}_e^e \tag{9.18}$$

gives

$$\dot{\boldsymbol{v}}_e^e = \boldsymbol{\alpha}^e \,, \tag{9.19}$$

where the control input $\boldsymbol{\alpha}^e$ has the meaning of an acceleration referred to the end-effector frame Σ_e. Hence, setting

$$\boldsymbol{\alpha}^e = \dot{\boldsymbol{v}}_d^e + \mathbf{K}_M^{-1}(\mathbf{K}_D \Delta \boldsymbol{v}_{de}^e + \boldsymbol{h}_\Delta^e - \boldsymbol{h}_e^e) \,, \tag{9.20}$$

the following expression can be found for the closed-loop system

$$\mathbf{K}_M \Delta \dot{\boldsymbol{v}}_{de}^e + \mathbf{K}_D \Delta \boldsymbol{v}_{de}^e + \boldsymbol{h}_\Delta^e = \boldsymbol{h}_e^e \,, \tag{9.21}$$

where $\mathbf{K}_M$ and $\mathbf{K}_D$ are 6×6 symmetric and positive-definite matrices, $\Delta\dot{\boldsymbol{v}}_{de}^e = \dot{\boldsymbol{v}}_d^e - \dot{\boldsymbol{v}}_e^e$, $\Delta\boldsymbol{v}_{de}^e = \boldsymbol{v}_d^e - \boldsymbol{v}_e^e$, $\dot{\boldsymbol{v}}_d^e$ and $\boldsymbol{v}_d^e$ are, respectively, the acceleration and the velocity of a desired frame Σ_d and $\boldsymbol{h}_\Delta^e$ is the elastic wrench (9.13); all the quantities are referred to the end-effector frame Σ_e.

The above equation describing the dynamic behavior of the controlled end-effector can be interpreted as a generalized mechanical impedance. The asymptotic stability of the equilibrium in the case $\boldsymbol{h}_e = \mathbf{0}$ can be proven by considering the Lyapunov function

$$V = \frac{1}{2}\Delta\boldsymbol{v}_{de}^{eT}\mathbf{K}_M\Delta\boldsymbol{v}_{de}^e + V_t + V_o \,, \tag{9.22}$$

where V_t and V_o are defined in (9.9) and (9.11), respectively, and whose time derivative along the trajectories of system (9.21) is the negative semidefinite function

$$\dot{V} = -\Delta\boldsymbol{v}_{de}^{eT}\mathbf{K}_D\Delta\boldsymbol{v}_{de}^e \,.$$

When $\boldsymbol{h}_e \neq \mathbf{0}$, a different asymptotically stable equilibrium can be found, corresponding to a nonnull displacement of the desired frame Σ_d with respect to the end-effector frame Σ_e. The new equilibrium is the solution of the equation $\boldsymbol{h}_\Delta = \boldsymbol{h}_e$.

In case Σ_d is constant, (9.21) has the meaning of a true 6-DOF mechanical impedance if $\mathbf{K}_M$ is chosen as

$$\mathbf{K}_M = \begin{pmatrix} m\mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{M} \end{pmatrix} \,,$$

where m is a mass and $\mathbf{M}$ is a 3×3 inertia tensor, and $\mathbf{K}_D$ is chosen as a block-diagonal matrix with 3×3 blocks. The physically equivalent system is a body of mass m with inertia tensor $\mathbf{M}$ with respect to a frame Σ_e attached to the body, subject to an external wrench $\boldsymbol{h}_e^e$. This body is connected to a virtual body attached to frame Σ_d through a 6-DOF ideal spring with stiffness matrix $\mathbf{K}_P$ and is subject to viscous forces and moments with damping $\mathbf{K}_D$. The function V in (9.22) represents the total energy of the body: the sum of the kinetic and potential elastic energy.

A block diagram of the resulting impedance control is sketched in Fig. 9.1. The impedance control computes the acceleration input as in (9.18) and (9.20) on the basis of the position and orientation feedback as well as the force and moment measurements. Then, the inverse dynamics control law computes the torques for the joint actuators $\boldsymbol{\tau} = \mathbf{J}^T\boldsymbol{h}_c$ with $\boldsymbol{h}_c$ in (9.16). This control scheme, in the absence of interaction, guarantees that the end-effector frame Σ_e asymptotically follows the desired frame Σ_d. In the presence of contact with the environment, a compliant dynamic behavior is imposed on the end-effector, according to the impedance (9.21), and the contact wrench is bounded at the expense of a finite position and orientation displacement between Σ_d and Σ_e. Differently from stiffness control, a force/torque sensor is required for the measurement of the contact force and moment.

In the case that the force/torque sensor is not available, the measure of the external wrench $\boldsymbol{h}_e$ cannot be used in the controller and thus the configuration-

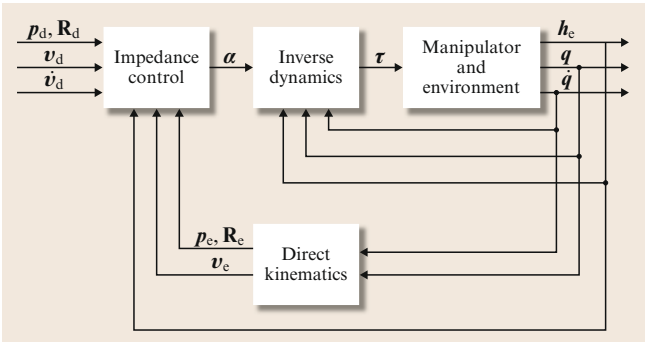


Fig. 9.1 Impedance control

Avidbots Ex1016
Page 247 of 2154

independent impedance behaviour (9.21) cannot be obtained anymore. However, a desired impedance behaviour can be still achieved with the control law

$$\boldsymbol{h}_{\mathrm{c}} = \boldsymbol{\Lambda}(\boldsymbol{q})\dot{\boldsymbol{v}}_{\mathrm{d}} + \boldsymbol{\Gamma}(\boldsymbol{q},\dot{\boldsymbol{q}})\boldsymbol{v}_{\mathrm{d}} + \mathbf{K}_{\mathrm{D}}\Delta\boldsymbol{v}_{\mathrm{de}} + \boldsymbol{h}_{\Delta} + \boldsymbol{\eta}(\boldsymbol{q}) \tag{9.23}$$

in place of (9.16), where $\mathbf{K}_{\mathrm{D}}$ is a 6×6 positive-definite matrix and $\boldsymbol{h}_{\Delta}$ is the elastic wrench (9.13). The resulting closed-loop equation is

$$\boldsymbol{\Lambda}(\boldsymbol{q})\Delta\dot{\boldsymbol{v}}_{\mathrm{de}} + (\boldsymbol{\Gamma}(\boldsymbol{q},\dot{\boldsymbol{q}}) + \mathbf{K}_{\mathrm{D}})\Delta\boldsymbol{v}_{\mathrm{de}} + \boldsymbol{h}_{\Delta} = \boldsymbol{h}_{\mathrm{e}},$$

representing an impedance behaviour which preserves the actual operational space inertia matrix $\boldsymbol{\Lambda}(\boldsymbol{q})$ of the robot. In the above equation, the centrifugal and Coriolis wrench $\boldsymbol{\Gamma}(\boldsymbol{q},\dot{\boldsymbol{q}})\Delta\boldsymbol{v}_{\mathrm{de}}$ is required to preserve the mechanical properties of a configuration-dependent inertia and to prove the stability similarly to (9.21). In case Σ_{d} is constant, control law (9.23) reduces to the stiffness control law (9.15).

Pioneering experiments on stiffness control and impedance control and without without force sensing are presented in VIDEO 684. VIDEO 686 reports experiments on impedance control based on a geometrically consistent active stiffness.

Implementation Issues

The selection of good impedance parameters ensuring a satisfactory behavior is not an easy task. In fact, the dynamics of the closed-loop system is different in free space and during interaction. The control objectives are different as well, since motion tracking and disturbance rejection must be ensured in free space, while, during the interaction, the main goal is achieving a suitable compliant dynamic behavior for the end-effector. Notice also that the dynamics of the controlled system during the interaction depends on the dynamics of the environment.

To gain insight into these problems, assume that the interaction of the end-effector with the environment can be approximated by that derived from an ideal 6-DOF spring connecting end-effector frame Σ_{e} to the environment frame Σ_{o}. Therefore, according to (9.8), the elastic wrench exerted by the end-effector on the environment, in the presence of an infinitesimal twist displacement of Σ_{e} with respect to Σ_{o}, can be computed as

$$\boldsymbol{h}_{\mathrm{e}}^{\mathrm{e}} = \mathbf{K}\delta\boldsymbol{x}_{\mathrm{eo}}^{\mathrm{e}}\,, \tag{9.24}$$

where Σ_{e} and Σ_{o} coincide at equilibrium and $\mathbf{K}$ is a stiffness matrix. The above model holds only in the presence of interaction, while the contact wrench is null when the end-effector moves in free space.

The disturbances acting on the robot manipulator and the unmodeled dynamics (joint friction, modeling errors, etc.) may be taken into account by introducing an additive term on the right-hand side of the dynamic model of the robot manipulator (9.2), corresponding to an equivalent disturbance wrench acting on the end-effector. This term produces an additive acceleration disturbance $\boldsymbol{\gamma}^{\mathrm{e}}$ on the right-hand side of (9.19). Therefore, using the control law (9.20), the following closed-loop impedance equation can be found

$$\mathbf{K}_{\mathrm{M}}\Delta\dot{\boldsymbol{v}}_{\mathrm{de}}^{\mathrm{e}} + \mathbf{K}_{\mathrm{D}}\Delta\boldsymbol{v}_{\mathrm{de}}^{\mathrm{e}} + \boldsymbol{h}_{\Delta}^{\mathrm{e}} = \boldsymbol{h}_{\mathrm{e}}^{\mathrm{e}} + \mathbf{K}_{\mathrm{M}}\boldsymbol{\gamma}^{\mathrm{e}}\,. \tag{9.25}$$

The tuning procedure for the impedance parameters can be set up starting from the linearized model that can be computed from (9.25) in the case of infinitesimal displacements, i. e.,

$$\begin{aligned}&\mathbf{K}_{\mathrm{M}}\delta\ddot{\boldsymbol{x}}_{\mathrm{de}}^{\mathrm{e}} + \mathbf{K}_{\mathrm{D}}\delta\dot{\boldsymbol{x}}_{\mathrm{de}}^{\mathrm{e}} + (\mathbf{K}_{\mathrm{P}} + \mathbf{K})\delta\boldsymbol{x}_{\mathrm{de}}^{\mathrm{e}}\\ &= \mathbf{K}\delta\boldsymbol{x}_{\mathrm{do}}^{\mathrm{e}} + \mathbf{K}_{\mathrm{M}}\boldsymbol{\gamma}^{\mathrm{e}}\,,\end{aligned} \tag{9.26}$$

where (9.24) and the equality $\delta\boldsymbol{x}_{\mathrm{eo}}^{\mathrm{e}} = -\delta\boldsymbol{x}_{\mathrm{de}}^{\mathrm{e}} + \delta\boldsymbol{x}_{\mathrm{do}}^{\mathrm{e}}$ have been used. The above equation is valid both for constrained ($\mathbf{K}\neq\mathbf{0}$) and for free motion ($\mathbf{K}=\mathbf{0}$).

It is evident that suitable dynamics of the position and orientation errors can be set by suitably choosing the matrix gains $\mathbf{K}_{\mathrm{M}}$, $\mathbf{K}_{\mathrm{D}}$, and $\mathbf{K}_{\mathrm{P}}$. This task is easier under the hypothesis that all the matrices are diagonal, resulting in a decoupled behavior for the six components of the infinitesimal twist displacement. In this case, the transient behavior of each component can be set, e.g., by assigning the natural frequency and damping ratio with the relations

$$\omega_{\mathrm{n}} = \sqrt{\frac{< k_{\mathrm{P}} + k}{k_{\mathrm{M}}}}\,, \quad \zeta = \frac{1}{2}\frac{k_{\mathrm{D}}}{\sqrt{k_{\mathrm{M}}(k_{\mathrm{P}}+k)}}\,.$$

Hence, if the gains are chosen so that a given natural frequency and damping ratio are ensured during the interaction (i. e., for $k\neq 0$), a smaller natural frequency with a higher damping ratio will be obtained when the end-effector moves in free space (i. e., for $k=0$). As for the steady-state performance, the end-effector error for the generic component is

$$\delta x_{\mathrm{de}} = \frac{k}{(k_{\mathrm{P}}+k)}\delta x_{\mathrm{do}} + \frac{k_{\mathrm{M}}}{k_{\mathrm{P}}+k}\boldsymbol{\gamma}$$

and the corresponding interaction force is

$$h = \frac{k_{\mathrm{P}}k}{k_{\mathrm{P}}+k}\delta x_{\mathrm{do}} - \frac{k_{\mathrm{M}}k}{k_{\mathrm{P}}+k}\boldsymbol{\gamma}\,.$$

Avidbots Ex1016
Page 248 of 2154

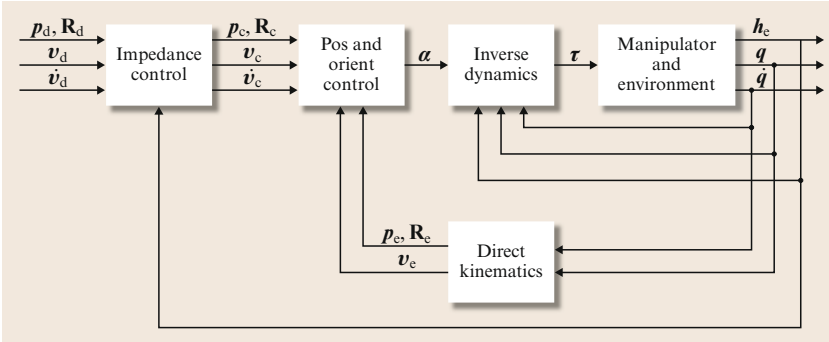


Fig. 9.2 Impedance control with inner motion control loop (admittance control)

The above relations show that, during interaction, the contact force can be made small at the expense of a large position error in steady state, as long as the active stiffness k_P is set low with respect to the stiffness of the environment k, and vice versa. However, both the contact force and the position error also depend on the external disturbance $\boldsymbol{\gamma}$; in particular, the lower k_P, the higher the influence of $\boldsymbol{\gamma}$ on both δx_{de} and h. Moreover, a low active stiffness k_P may result in a large position error also in the absence of interaction (i. e., when $k = 0$).

Admittance Control

A solution to this drawback can be devised by separating motion control from impedance control as follows. The motion control action is purposefully made stiff so as to enhance disturbance rejection but, rather than ensuring tracking of the desired end-effector position and orientation, it ensures tracking of a reference position and orientation resulting from the impedance control action. In other words, the desired position and orientation, together with the measured contact wrench, are input to the impedance equation which, via a suitable integration, generates the position and orientation to be used as a reference for the motion control.

To implement this solution, it is worth introducing a reference frame other than the desired frame Σ_d. This frame is referred to as the *compliant frame* Σ_c, and is specified by the quantities $\boldsymbol{p}_c$, $\mathbf{R}_c$, $\boldsymbol{v}_c$, and $\dot{\boldsymbol{v}}_c$ that are computed from $\boldsymbol{p}_d$, $\mathbf{R}_d$, $\boldsymbol{v}_d$, and $\dot{\boldsymbol{v}}_d$ and the measured wrench $\boldsymbol{h}_c$, by integrating the equation

$$\mathbf{K}_M \Delta \dot{\boldsymbol{v}}_{dc}^c + \mathbf{K}_D \Delta \boldsymbol{v}_{dc}^c + \boldsymbol{h}_\Delta^c = \boldsymbol{h}^c \,, \tag{9.27}$$

where $\boldsymbol{h}_\Delta^c$ is the elastic wrench in the presence of a finite displacement between the desired frame Σ_d and the compliant frame Σ_c. Then, a motion control strategy, based on inverse dynamics, is designed so that the end-effector frame Σ_e is taken to coincide with the compliant frame Σ_c. To guarantee the stability of the overall system, the bandwidth of the motion controller should be higher than the bandwidth of the impedance controller.

A block diagram of the resulting scheme is sketched in Fig. 9.2. It is evident that, in the absence of interaction, the compliant frame Σ_c coincides with the desired frame Σ_d and the dynamics of the position and orientation error, as well as the disturbance rejection capabilities, depend only on the gains of the inner motion control loop. On the other hand, the dynamic behavior in the presence of interaction is imposed by the impedance gains (9.27).

The control scheme of Fig. 9.2 is also known as *admittance control* because, in (9.27), the measured force (the input) is used to compute the motion of the compliant frame (the output), given the motion of the desired frame; a mapping with a force as input and a position or velocity as output corresponds to a mechanical *admittance*. Vice versa, (9.21), mapping the end-effector displacement (the input) from the desired motion trajectory into the contact wrench (the output), has the meaning of a mechanical *impedance*. Experiments on admittance control are reported in VIDEO 685.

Simplified Schemes

The inverse dynamics control is model based and requires modification of current industrial robot controllers, which are usually equipped with independent proportional–integral (PI) joint velocity controllers with very high bandwidth. These controllers are able to decouple the robot dynamics to a large extent, especially in the case of slow motion, and to mitigate the effects of external forces on the manipulator motion if the environment is sufficiently compliant. Hence, the closed-loop dynamics of the controlled robot can be approximated by

$$\dot{\boldsymbol{q}} = \dot{\boldsymbol{q}}_r$$

in joint space, or equivalently

$$\dot{\boldsymbol{v}}_e = \boldsymbol{v}_r \tag{9.28}$$

in the operational space, where $\dot{\boldsymbol{q}}_r$ and $\boldsymbol{v}_r$ are the control signals for the inner velocity motion loop generated by

Avidbots Ex1016
Page 249 of 2154

a suitably designed outer control loop. These control signals are related by

$$\dot{\boldsymbol{q}}_\mathrm{r} = \mathbf{J}^{-1}(\boldsymbol{q})\boldsymbol{v}_\mathrm{r} \,.$$

The velocity $\boldsymbol{v}_\mathrm{r}$, corresponding to a *velocity-resolved control*, can be computed as

$$\boldsymbol{v}_\mathrm{r}^\mathrm{e} = \boldsymbol{v}_\mathrm{d}^\mathrm{e} + \mathbf{K}_\mathrm{D}^{-1}\left(\boldsymbol{h}_\Delta^\mathrm{e} - \boldsymbol{h}_\mathrm{e}^\mathrm{e}\right) \,,$$

where the control input has been referred to the end-effector frame, $\mathbf{K}_\mathrm{D}$ is a 6×6 positive-definite matrix and $\boldsymbol{h}_\Delta$ is the elastic wrench (9.13) with stiffness matrix $\mathbf{K}_\mathrm{P}$. The resulting closed-loop equation is

$$\mathbf{K}_\mathrm{D}\Delta\boldsymbol{v}_\mathrm{de}^\mathrm{e} + \boldsymbol{h}_\Delta^\mathrm{e} = \boldsymbol{h}_\mathrm{e}^\mathrm{e} \,,$$

corresponding to a compliant behavior of the end-effector characterized by a damping $\mathbf{K}_\mathrm{D}$ and a stiffness $\mathbf{K}_\mathrm{P}$. In the case $\mathbf{K}_\mathrm{P} = \mathbf{0}$, the resulting scheme is known as *damping control*.

Alternatively, an admittance-type control scheme can be adopted, where the motion of a compliant frame Σ_c can be computed as the solution of the differential equation

$$\mathbf{K}_\mathrm{D}\Delta\boldsymbol{v}_\mathrm{dc}^\mathrm{c} + \boldsymbol{h}_\Delta^\mathrm{c} = \boldsymbol{h}_\mathrm{e}^\mathrm{c}$$

in terms of the position $\boldsymbol{p}_\mathrm{c}$, orientation $\mathbf{R}_\mathrm{c}$, and velocity twist $\boldsymbol{v}_\mathrm{c}$, where the inputs are the motion variables of the desired frame Σ_d and the contact wrench $\boldsymbol{h}_\mathrm{e}^\mathrm{c}$. The motion variables of Σ_c are then input to an inner position and velocity controller. In the case $\mathbf{K}_\mathrm{D} = \mathbf{0}$, the resulting scheme is known as *compliance control*.

9.3 Interaction Tasks

Indirect force control does not require explicit knowledge of the environment, although to achieve a satisfactory dynamic behavior the control parameters have to be tuned for a particular task. On the other hand, a model of the interaction task is required for the synthesis of direct force control algorithms.

An interaction task is characterized by complex contact situations between the manipulator and the environment. To guarantee proper task execution, it is necessary to have an analytical description of the interaction force and moment, which is very demanding from a modeling viewpoint.

A real contact situation is a naturally distributed phenomenon in which the local characteristics of the contact surfaces as well as the global dynamics of the manipulator and environment are involved. In detail:

- The environment imposes kinematic constraints on the end-effector motion, due to one or more contacts of different type, and a reaction wrench arises when the end-effector tends to violate the constraints (e.g., the case of a robot sliding a rigid tool on a frictionless rigid surface).
- The end-effector, while being subject to kinematic constraints, may also exert a dynamic wrench on the environment, in the presence of environment dynamics (e.g., the case of a robot turning a crank, when the crank dynamics is relevant, or a robot pushing against a compliant surface).
- The contact wrench may depend on the structural compliance of the robot, due to the finite stiffness of the joints and links of the manipulator, as well as of the wrist force/torque sensor or of the tool (e.g., an end-effector mounted on an RCC device).
- Local deformation of the contact surfaces may occur during the interaction, producing distributed contact areas (e.g., the case of a soft contact surface of the tool or of the environment).
- Static and dynamic friction may occur in the case of non ideally smooth contact surfaces.

The design of the interaction control and the performance analysis are usually carried out under simplifying assumptions. The following two cases are considered:

1. The robot and the environment are perfectly rigid and purely kinematics constraints are imposed by the environment,
2. The robot is perfectly rigid, all the compliance in the system is localized in the environment, and the contact wrench is approximated by a linear elastic model.

In both cases, frictionless contact is assumed. It is obvious that these situations are only ideal. However, the robustness of the control should be able to cope with situations where some of the ieal assumptions are relaxed. In that case the control laws may be adapted to deal with nonideal characteristics.

9.3.1 Rigid Environment

The kinematic constraints imposed by the environment can be represented by a set of equations that the

variables describing the end-effector position and orientation must satisfy; since these variables depend on the joint variables through the direct kinematic equations, the constraint equations can also be expressed in the joint space as

$$\boldsymbol{\phi}(\boldsymbol{q}) = \mathbf{0}\,. \tag{9.29}$$

The vector $\boldsymbol{\phi}$ is an $m\times 1$ function, with $m < n$, where n is the number of joints of the manipulator, assumed to be nonredundant; without loss of generality, the case $n = 6$ is considered. Constraints of the form (9.29), involving only the generalized coordinates of the system, are known as holonomic constraints. The case of time-varying constraints of the form $\boldsymbol{\phi}(\boldsymbol{q}, t) = \mathbf{0}$ is not considered here but can be analyzed in a similar way. Moreover, only bilateral constraints expressed by equalities of the form (9.29) are of concern; this means that the end-effector always keeps contact with the environment. The analysis presented here is known as *kinetostatic analysis*.

It is assumed that the vector (9.29) is twice differentiable and that its m components are linearly independent at least locally in a neighborhood of the operating point. Hence, differentiation of (9.29) yields

$$\mathbf{J}_{\phi}(\boldsymbol{q})\dot{\boldsymbol{q}} = \mathbf{0}\,, \tag{9.30}$$

where $\mathbf{J}_{\phi}(\boldsymbol{q}) = \partial\boldsymbol{\phi}/\partial\boldsymbol{q}$ is the $m\times 6$ Jacobian of $\boldsymbol{\phi}(\boldsymbol{q})$, known as the *constraint Jacobian*. By virtue of the above assumption, $\mathbf{J}_{\phi}(\boldsymbol{q})$ is of rank m at least locally in a neighborhood of the operating point.

In the absence of friction, the generalized interaction forces are represented by a reaction wrench that tends to violate the constraints. This end-effector wrench produces reaction torques at the joints that can be computed using the principle of virtual work as

$$\boldsymbol{\tau}_{\mathrm{e}} = \mathbf{J}_{\phi}^{\mathrm{T}}(\boldsymbol{q})\boldsymbol{\lambda}\,,$$

where $\boldsymbol{\lambda}$ is an $m\times 1$ vector of Lagrange multipliers. The end-effector wrench corresponding to $\boldsymbol{\tau}_{\mathrm{e}}$ can be computed as

$$\boldsymbol{h}_{\mathrm{e}} = \mathbf{J}^{-\mathrm{T}}(\boldsymbol{q})\boldsymbol{\tau}_{\mathrm{e}} = \mathbf{S}_{\mathrm{f}}(\boldsymbol{q})\boldsymbol{\lambda}\,, \tag{9.31}$$

where

$$\mathbf{S}_{\mathrm{f}} = \mathbf{J}^{-\mathrm{T}}(\boldsymbol{q})\mathbf{J}_{\phi}^{\mathrm{T}}(\boldsymbol{q})\,. \tag{9.32}$$

From (9.31) it follows that $\boldsymbol{h}_{\mathrm{e}}$ belongs to the m-dimensional vector space spanned by the columns of the $6\times m$ matrix $\mathbf{S}_{\mathrm{f}}$. The inverse of the linear transformation (9.31) is computed as

$$\boldsymbol{\lambda} = \mathbf{S}_{\mathrm{f}}^{\dagger}(\boldsymbol{q})\boldsymbol{h}_{\mathrm{e}}\,, \tag{9.33}$$

where $\mathbf{S}_{\mathrm{f}}^{\dagger}$ denotes a weighted pseudoinverse of the matrix $\mathbf{S}_{\mathrm{f}}$, i. e.,

$$\mathbf{S}_{\mathrm{f}}^{\dagger} = \left(\mathbf{S}_{\mathrm{f}}^{\mathrm{T}}\mathbf{W}\mathbf{S}_{\mathrm{f}}\right)^{-1}\mathbf{S}_{\mathrm{f}}^{\mathrm{T}}\mathbf{W}\,, \tag{9.34}$$

where $\mathbf{W}$ is a suitable weighting matrix.

Notice that, while the range space of the matrix $\mathbf{S}_{\mathrm{f}}$ in (9.32) is uniquely defined by the geometry of the contact, the matrix $\mathbf{S}_{\mathrm{f}}$ itself is not unique; also, the constraint equations (9.29), the corresponding Jacobian $\mathbf{J}_{\phi}$ as well as the pseudoinverse $\mathbf{S}_{\mathrm{f}}^{\dagger}$ and the vector $\boldsymbol{\lambda}$ are not uniquely defined.

In general, the physical units of measure of the elements of $\boldsymbol{\lambda}$ are not homogeneous and the columns of the matrix $\mathbf{S}_{\mathrm{f}}$, as well as of the matrix $\mathbf{S}_{\mathrm{f}}^{\dagger}$, do not necessarily represent homogeneous entities. This may produce invariance problems in the transformation (9.33) if $\boldsymbol{h}_{\mathrm{e}}$ represents a measured wrench that is subject to disturbances and, as a result, may have components outside the range space of $\mathbf{S}_{\mathrm{f}}$. If a physical unit or a reference frame is changed, the matrix $\mathbf{S}_{\mathrm{f}}$ undergoes a transformation; however, the result of (9.33) with the transformed pseudoinverse in general depends on the adopted physical units or on the reference frame. The reason is that the pseudoinverse is the weighted least-squares solution of a minimization problem based on the norm of the vector $\boldsymbol{h}_{\mathrm{e}} - \mathbf{S}_{\mathrm{f}}(\boldsymbol{q})\boldsymbol{\lambda}$, and the invariance can be guaranteed only if a physically consistent norm of this vector is used. In the ideal case that $\boldsymbol{h}_{\mathrm{e}}$ is in the range space of $\mathbf{S}_{\mathrm{f}}$, there is a unique solution for $\boldsymbol{\lambda}$ in (9.33), regardless of the weighting matrix, and hence the invariance problem does not appear.

A possible solution consists of choosing $\mathbf{S}_{\mathrm{f}}$ so that its columns represent linearly independent wrenches. This implies that (9.31) gives $\boldsymbol{h}_{\mathrm{e}}$ as a linear combination of wrenches and $\boldsymbol{\lambda}$ is a dimensionless vector. A physically consistent norm on the wrench space can be defined based on the quadratic form $\boldsymbol{h}_{\mathrm{e}}^{\mathrm{T}}\mathbf{K}^{-1}\boldsymbol{h}_{\mathrm{e}}$, which has the meaning of an elastic energy if $\mathbf{K}$ is a positive-definite matrix corresponding to a stiffness. Hence, the choice $\mathbf{W} = \mathbf{K}^{-1}$ can be made for the weighting matrix of the pseudoinverse.

Notice that, for a given $\mathbf{S}_{\mathrm{f}}$, the constraint Jacobian can be computed from (9.32) as $\mathbf{J}_{\phi}(\boldsymbol{q}) = \mathbf{S}_{\mathrm{f}}^{\mathrm{T}}\mathbf{J}(\boldsymbol{q})$; moreover, the constraint equations can be derived by integrating (9.30).

Using (9.1) and (9.32), the equality (9.30) can be rewritten in the form

$$\mathbf{J}_{\phi}(\boldsymbol{q})\mathbf{J}^{-1}(\boldsymbol{q})\mathbf{J}(\boldsymbol{q})\dot{\boldsymbol{q}} = \mathbf{S}_{\mathrm{f}}^{\mathrm{T}}\boldsymbol{v}_{\mathrm{e}} = \mathbf{0}\,, \tag{9.35}$$

which, by virtue of (9.31), is equivalent to

$$\boldsymbol{h}_{\mathrm{e}}^{\mathrm{T}}\boldsymbol{v}_{\mathrm{e}} = 0\,. \tag{9.36}$$

Avidbots Ex1016
Page 251 of 2154

Equation (9.36) represents the kinetostatic relationship, known as *reciprocity*, between the ideal reaction wrench $\boldsymbol{h}_e$ (belonging to the so-called *force-controlled* subspace) and the end-effector twist that obeys the constraints (belonging to the so-called *velocity-controlled* subspace). The concept of reciprocity, expressing the physical fact that, in the hypothesis of rigid and frictionless contact, the wrench does not cause any work against the twist, is often confused with the concept of *orthogonality*, which makes no sense in this case because twists and wrenches belong to different spaces.

Equations ((9.35)) and (9.36) imply that the velocity-controlled subspace is the reciprocal complement of the m-dimensional force-controlled subspace, identified by the range of matrix $\mathbf{S}_f$. Hence, the dimension of the velocity-controlled subspace is $6-m$ and a $6\times(6-m)$ matrix $\mathbf{S}_v$ can be defined, whose columns span the velocity-controlled subspace, i. e.,

$$\boldsymbol{v}_e = \mathbf{S}_v(\boldsymbol{q})\boldsymbol{\nu}\,, \tag{9.37}$$

where $\boldsymbol{\nu}$ is a suitable $(6-m)\times 1$ vector. From (9.35) and (9.37) the following equality holds

$$\mathbf{S}_f^T(\boldsymbol{q})\mathbf{S}_v(\boldsymbol{q}) = \mathbf{0}\,; \tag{9.38}$$

moreover, the inverse of the linear transformation (9.37) can be computed as

$$\boldsymbol{\nu} = \mathbf{S}_v^\dagger(\boldsymbol{q})\boldsymbol{v}_e\,, \tag{9.39}$$

where $\mathbf{S}_v^\dagger$ denotes a suitable weighted pseudoinverse of the matrix $\mathbf{S}_v$, computed as in (9.34).

Notice that, as for the case of $\mathbf{S}_f$, although the range space of the matrix $\mathbf{S}_v$ is uniquely defined, the choice of the matrix $\mathbf{S}_v$ itself is not unique. Moreover, the columns of $\mathbf{S}_v$ are not necessarily twists and the scalar $\boldsymbol{\nu}$ may have nonhomogeneous physical dimensions. However, in order to avoid invariance problems analogous to that considered for the case of $\mathbf{S}_f$, it is convenient to select the columns of $\mathbf{S}_v$ as twists so that the vector $\boldsymbol{\nu}$ is dimensionless; moreover, the weighting matrix used to compute the pseudoinverse in (9.39) can be set as $\mathbf{W}=\mathbf{M}$, being $\mathbf{M}$ a 6×6 inertia matrix; this corresponds to defining a norm in the space of twists based on the kinetic energy. It is worth observing that the transformation matrices of twists and wrenches, corresponding to a change of reference frame, are different; however, if twists are defined with angular velocity on top and translational velocity on bottom, then their transformation matrix is the same as for wrenches.

The matrix $\mathbf{S}_v$ may also have an interpretation in terms of Jacobians, as for $\mathbf{S}_f$ in (9.32). Due to the presence of m independent holonomic constraints (9.29), the configuration of the robot in contact with the environment can be described in terms of a $(6-m)\times 1$ vector $\boldsymbol{r}$ of independent variables. From the implicit function theorem, this vector can be defined as

$$\boldsymbol{r} = \boldsymbol{\psi}(\boldsymbol{q})\,, \tag{9.40}$$

where $\boldsymbol{\psi}(\boldsymbol{q})$ is any $(6-m)\times 1$ twice-differentiable vector function such that the m components of $\boldsymbol{\phi}(\boldsymbol{q})$ and the $n-m$ components of $\boldsymbol{\psi}(\boldsymbol{q})$ are linearly independent at least locally in a neighborhood of the operating point. This means that the mapping (9.40), together with the constraint (9.29), is locally invertible, with inverse defined as

$$\boldsymbol{q} = \boldsymbol{\rho}(\boldsymbol{r})\,, \tag{9.41}$$

where $\boldsymbol{\rho}(\boldsymbol{r})$ is a 6×1 twice-differentiable vector function. Equation (9.41) explicitly provides all the joint vectors which satisfy the constraint (9.29). Moreover, the joint velocity vectors that satisfy (9.30) can be computed as

$$\dot{\boldsymbol{q}} = \mathbf{J}_\rho(\boldsymbol{r})\dot{\boldsymbol{r}}\,,$$

where $\mathbf{J}_\rho(\boldsymbol{r}) = \partial\boldsymbol{\rho}/\partial\boldsymbol{r}$ is a $6\times(6-m)$ full-rank Jacobian matrix. Therefore, the following equality holds

$$\mathbf{J}_\phi(\boldsymbol{q})\mathbf{J}_\rho(\boldsymbol{r}) = \mathbf{0}\,,$$

which can be interpreted as a reciprocity condition between the subspace of the reaction torques spanned by the columns of the matrix $\mathbf{J}_\phi^T$ and the subspace of the constrained joint velocities spanned by the columns of the matrix $\mathbf{J}_\rho$.

Rewriting the above equation as

$$\mathbf{J}_\phi(\boldsymbol{q})\mathbf{J}(\boldsymbol{q})^{-1}\mathbf{J}(\boldsymbol{q})\mathbf{J}_\rho(\boldsymbol{r}) = \mathbf{0}\,,$$

and taking into account (9.32) and (9.38), the matrix $\mathbf{S}_v$ can be expressed as

$$\mathbf{S}_v = \mathbf{J}(\boldsymbol{q})\mathbf{J}_\rho(\boldsymbol{r})\,, \tag{9.42}$$

which, by virtue of (9.40) and (9.41), it can be equivalently expressed as a function of either $\boldsymbol{q}$ or $\boldsymbol{r}$.

The matrices $\mathbf{S}_f$ and $\mathbf{S}_v$, and their pseudoinverse $\mathbf{S}_f^\dagger$ and $\mathbf{S}_v^\dagger$ are known as selection matrices. They play a fundamental role for the task specification, i. e., the specification of the desired end-effector motion and interaction forces and moments, as well as for the control synthesis.

Avidbots Ex1016
Page 252 of 2154

9.3.2 Compliant Environment

In many applications, the interaction wrench between the end-effector and a compliant environment can be approximated by an ideal elastic model of the form (9.24). However, since the stiffness matrix $\mathbf{K}$ is positive definite, this model describes a fully constrained case, when the environment deformation coincides with the infinitesimal twist displacement of the end-effector. In general, however, the end-effector motion is only partially constrained by the environment and this situation can be modeled by introducing a suitable positive-semidefinite stiffness matrix.

The stiffness matrix describing the partially constrained interaction between the end-effector and the environment can be computed by modeling the environment as a couple of rigid bodies, S and O, connected through an ideal 6-DOF spring of compliance $\mathbf{C} = \mathbf{K}^{-1}$. Body S is attached to a frame Σ_s and is in contact with the end-effector; body O is attached to a frame Σ_o which, at equilibrium, coincides with frame Σ_s. The environment deformation about the equilibrium, in the presence of a wrench $\boldsymbol{h}_s$, is represented by the infinitesimal twist displacement $\delta\boldsymbol{x}_{so}$ between frames Σ_s and Σ_o, which can be computed as

$$\delta\boldsymbol{x}_{so} = \mathbf{C}\,\boldsymbol{h}_s \,. \tag{9.43}$$

All the quantities hereafter are referred to frame Σ_s but the superscript s is omitted for brevity.

For the considered contact situation, the end-effector twist does not completely belong to the ideal velocity subspace, corresponding to a rigid environment, because the environment can deform. Therefore, the infinitesimal twist displacement of the end-effector frame Σ_e with respect to Σ_o can be decomposed as

$$\delta\boldsymbol{x}_{eo} = \delta\boldsymbol{x}_v + \delta\boldsymbol{x}_f \,, \tag{9.44}$$

where $\delta\boldsymbol{x}_v$ is the end-effector infinitesimal twist displacement in the velocity-controlled subspace, defined as the $6-m$ reciprocal complement of the force-controlled subspace, while $\delta\boldsymbol{x}_f$ is the end-effector infinitesimal twist displacement corresponding to the environment deformation. Hence,

$$\delta\boldsymbol{x}_v = \mathbf{P}_v\delta\boldsymbol{x}_{eo} \,, \tag{9.45}$$

$$\delta\boldsymbol{x}_f = (\mathbf{I}-\mathbf{P}_v)\delta\boldsymbol{x}_{eo} = (\mathbf{I}-\mathbf{P}_v)\delta\boldsymbol{x}_{so} \,, \tag{9.46}$$

where $\mathbf{P}_v = \mathbf{S}_v\mathbf{S}_v^{\dagger}$ and $\mathbf{S}_v$ and $\mathbf{S}_v^{\dagger}$ are defined as in the rigid environment case. Being $\mathbf{P}_v\mathbf{P}_v = \mathbf{P}_v$, the matrix $\mathbf{P}_v$ is a projection matrix that filters out all the end-effector twists (and infinitesimal twist displacements) that are not in the range space of $\mathbf{S}_v$. Moreover, $\mathbf{I}-\mathbf{P}_v$ is a projection matrix that filters out all the end-effector twists (and infinitesimal twist displacements) that are in the range space of $\mathbf{S}_v$. The twists $\mathbf{P}_v\boldsymbol{v}$ are denoted as *twists of freedom* while the twists $(\mathbf{I}-\mathbf{P}_v)\boldsymbol{v}$ are denoted as *twists of constraint*.

In the hypothesis of frictionless contact, the interaction wrench between the end-effector and the environment is restricted to a force-controlled subspace defined by the m-dimensional range space of a matrix $\mathbf{S}_f$, as for the rigid environment case, i. e.,

$$\boldsymbol{h}_e = \mathbf{S}_f\boldsymbol{\lambda} = \boldsymbol{h}_s \,, \tag{9.47}$$

where $\boldsymbol{\lambda}$ is an $m\times 1$ dimensionless vector. Premultiplying both sides of (9.44) by $\mathbf{S}_f^{\mathrm{T}}$ and using (9.43), (9.45), (9.46), and (9.47) yields

$$\mathbf{S}_f^{\mathrm{T}}\delta\boldsymbol{x}_{eo} = \mathbf{S}_f^{\mathrm{T}}\mathbf{C}\,\mathbf{S}_f\boldsymbol{\lambda} \,,$$

where the identity $\mathbf{S}_f^{\mathrm{T}}\mathbf{P}_v = \mathbf{0}$ has been exploited. Therefore, the following elastic model can be found

$$\boldsymbol{h}_e = \mathbf{S}_f\boldsymbol{\lambda} = \mathbf{K}'\delta\boldsymbol{x}_{eo} \,, \tag{9.48}$$

where $\mathbf{K}' = \mathbf{S}_f(\mathbf{S}_f^{\mathrm{T}}\mathbf{C}\mathbf{S}_f)^{-1}\mathbf{S}_f^{\mathrm{T}}$ is the positive-semidefinite stiffness matrix corresponding to the partially constrained interaction.

If the compliance matrix $\mathbf{C}$ is adopted as a weighting matrix for the computation of $\mathbf{S}_f^{\dagger}$, then $\mathbf{K}'$ can be expressed as

$$\mathbf{K}' = \mathbf{P}_f\mathbf{K} \,, \tag{9.49}$$

where $\mathbf{P}_f = \mathbf{S}_f\mathbf{S}_f^{\dagger}$. Being $\mathbf{P}_f\mathbf{P}_f = \mathbf{P}_f$, matrix $\mathbf{P}_f$ is a projection matrix that filters out all the end-effector wrenches that are not in the range space of $\mathbf{S}_f$.

The compliance matrix for the partially constrained interaction cannot be computed as the inverse of $\mathbf{K}'$, since this matrix is of rank $m<6$. However, using (9.46), (9.43), and (9.47), the following equality can be found

$$\delta\boldsymbol{x}_f = \mathbf{C}'\boldsymbol{h}_e \,,$$

where the matrix

$$\mathbf{C}' = (\mathbf{I}-\mathbf{P}_v)\mathbf{C} \,, \tag{9.50}$$

of rank $6-m$, is positive semidefinite. If the stiffness matrix $\mathbf{K}$ is adopted as a weighting matrix for the computation of $\mathbf{S}_v^{\dagger}$, then the matrix $\mathbf{C}'$ has the noticeable expression $\mathbf{C}' = \mathbf{C}-\mathbf{S}_v(\mathbf{S}_v^{\mathrm{T}}\mathbf{K}\,\mathbf{S}_v)^{-1}\mathbf{S}_v^{\mathrm{T}}$, showing that $\mathbf{C}'$ is symmetric.

Avidbots Ex1016
Page 253 of 2154

9.3.3 Task Specification

An interaction task can be assigned in terms of a desired end-effector wrench $\boldsymbol{h}_\mathrm{d}$ and twist $\boldsymbol{v}_\mathrm{d}$. In order to be consistent with the constraints, these vectors must lie in the force- and velocity-controlled subspaces, respectively. This can be guaranteed by specifying vectors $\boldsymbol{\lambda}_\mathrm{d}$ and $\boldsymbol{\nu}_\mathrm{d}$ and computing $\boldsymbol{h}_\mathrm{d}$ and $\boldsymbol{v}_\mathrm{d}$ as

$$\boldsymbol{h}_\mathrm{d} = \mathbf{S}_\mathrm{f}\boldsymbol{\lambda}_\mathrm{d}, \quad \boldsymbol{v}_\mathrm{d} = \mathbf{S}_\mathrm{v}\boldsymbol{\nu}_\mathrm{d} ,$$

where $\mathbf{S}_\mathrm{f}$ and $\mathbf{S}_\mathrm{v}$ have to be suitably defined on the basis of the geometry of the task, and so that invariance with respect to the choice of the reference frame and change of physical units is guaranteed.

Many robotic tasks have a set of orthogonal reference frames in which the task specification is very easy and intuitive. Such frames are called *task frames* or *compliance frames*. An interaction task can be specified by assigning a desired force/torque or a desired linear/angular velocity along/about each of the frame axes. The desired quantities are known as *artificial constraints* because they are imposed by the controller; these constraints, in the case of rigid contact, are complementary to those imposed by the environment, known as *natural constraints*.

Some examples of task frame definition and task specification are given below.

Peg-in-Hole

The goal of this task is to push the peg into the hole while avoiding wedging and jamming. The peg has two degrees of motion freedom, hence the dimension of the velocity-controlled subspace is $6-m=2$, while the dimension of the force-controlled subspace is $m=4$. The task frame can be chosen as shown in Fig. 9.3 and the task can be achieved by assigning the following desired forces and torques:

- Zero forces along the x_t and y_t axes
- Zero torques about the x_t and y_t axes,

and the desired velocities:

- A nonzero linear velocity along the z_t-axis
- An arbitrary angular velocity about the z_t-axis.

The task continues until a *large* reaction force in the z_t direction is measured, indicating that the peg has hit the bottom of the hole, not represented in the figure. Hence, the matrices $\mathbf{S}_\mathrm{f}$ and $\mathbf{S}_\mathrm{v}$ can be chosen as

$$\mathbf{S}_\mathrm{f} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \qquad \mathbf{S}_\mathrm{v} = \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \end{pmatrix},$$

where the columns of $\mathbf{S}_\mathrm{f}$ have the dimensions of wrenches and those of $\mathbf{S}_\mathrm{v}$ have the dimensions of twists, defined in the task frame, and they transform accordingly when changing the reference frame. The task frame can be chosen either attached to the end-effector or to the environment.

Turning a Crank

The goal of this task is turning a crank with an idle handle. The handle has two degrees of motion freedom, corresponding to the rotation about the z_t-axis and to the rotation about the rotation axis of the crank. Hence the dimension of the velocity-controlled subspace is $6-m=2$, while the dimension of the force-controlled subspace is $m=4$. The task frame can be assumed as in the Fig. 9.4, attached to the crank. The task can be achieved by assigning the following desired forces and torques:

- Zero forces along the x_t and z_t axes
- Zero torques about the x_t and y_t axes.

and the following desired velocities:

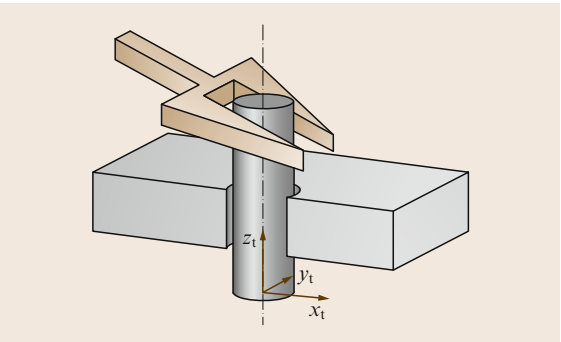


Fig. 9.3 Insertion of a cylindrical peg into a hole

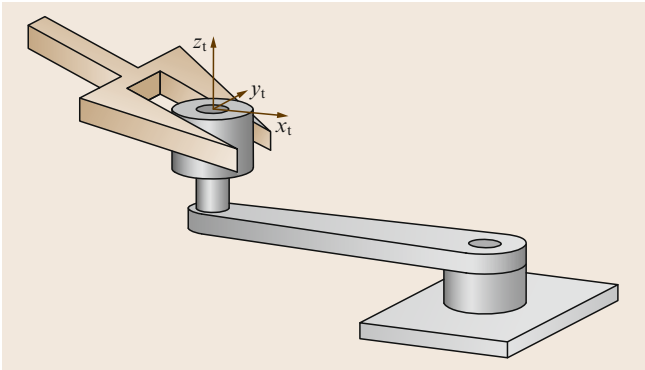


Fig. 9.4 Turning a crank with an idle handle

Avidbots Ex1016
Page 254 of 2154

- A nonzero linear velocity along the y_t-axis
- An arbitrary angular velocity about the z_t-axis.

Hence, the matrices $\mathbf{S}_f$ and $\mathbf{S}_v$ can be chosen as

$$\mathbf{S}_f = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \quad \mathbf{S}_v = \begin{pmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \end{pmatrix},$$

referred to the task frame. In this case, the task frame is fixed with respect to the crank, but in motion with respect both the end-effector frame (fixed to the handle) and to the base frame of the robot. Hence, the matrices $\mathbf{S}_f$ and $\mathbf{S}_v$ are time variant when referred either to the end-effector frame or to the base frame.

Sliding a Block on a Planar Elastic Surface

The goal of this task is to slide a prismatic block over a planar surface along the x_t-axis, while pushing with a given force against the elastic planar surface. The object has three degrees of motion freedom, hence the dimension of the velocity-controlled subspace is $6 - m = 3$ while the dimension of the force-controlled subspace is $m = 3$. The task frame can be chosen attached to the environment, as shown in Fig. 9.5, and the task can be achieved by assigning the desired velocities:

- A nonzero velocity along the x_t-axis
- A zero velocity along the y_t-axis
- A zero angular velocity about the z_t-axis,

and the desired forces and torques:

- A nonzero force along the z_t-axis
- Zero torques about the x_t and y_t axes.

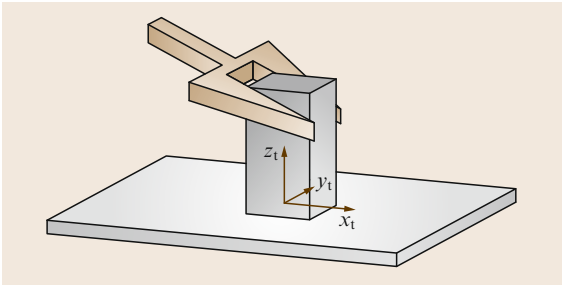


Fig. 9.5 Sliding of a prismatic object on a planar elastic surface

Hence, the matrices $\mathbf{S}_f$ and $\mathbf{S}_v$ can be chosen as

$$\mathbf{S}_f = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}, \quad \mathbf{S}_v = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

The elements of the 6×6 stiffness matrix $\mathbf{K}'$, corresponding to the partially constrained interaction of the end-effector with the environment, are all zero except those of the 3×3 principal minor $\mathbf{K}'_m$ formed by rows $3, 4, 5$ and columns $3, 4, 5$ of $\mathbf{K}'$, which can be computed as

$$\mathbf{K}'_m = \begin{pmatrix} c_{3,3} & c_{3,4} & c_{3,5} \\ c_{4,3} & c_{4,4} & c_{4,5} \\ c_{5,3} & c_{5,4} & c_{5,5} \end{pmatrix}^{-1},$$

where $c_{i,j} = c_{j,i}$ are the elements of the compliance matrix $\mathbf{C}$.

General Contact Model

The task frame concept has proven to be very useful for the specification of a variety of practical robotic tasks. However, it only applies to task geometries with limited complexity, for which separate control modes can be assigned independently to three pure translational and three pure rotational directions along the axes of a single frame. For more complex situations, as in the case of multiple-point contact, a task frame may not exist and more complex contact models have to be adopted. A possible solution is represented by the *virtual contact manipulator* model, where each individual contact is modeled by a virtual kinematic chain between the manipulated object and the environment, giving the manipulated object (instantaneously) the same motion freedom as the contact. The velocities and force kinematics of the parallel manipulator, formed by the virtual manipulators of all individual contacts, can be derived using the standard kinematics procedures of real manipulators and allow the construction of bases for the twist and wrench spaces of the total motion constraint.

A more general approach, known as *constraint-based task specification* opens up new applications involving complex geometries and/or the use of multiple sensors (force/torque, distance, visual sensors) for controlling different directions in space simultaneously. The concept of task frame is extended to that of multiple *feature frames*. Each of the feature frames makes it possible to model part of the task geometry using translational and rotational directions along the axes of a frame; also, part of the constraints is specified in each

Avidbots Ex1016
Page 255 of 2154

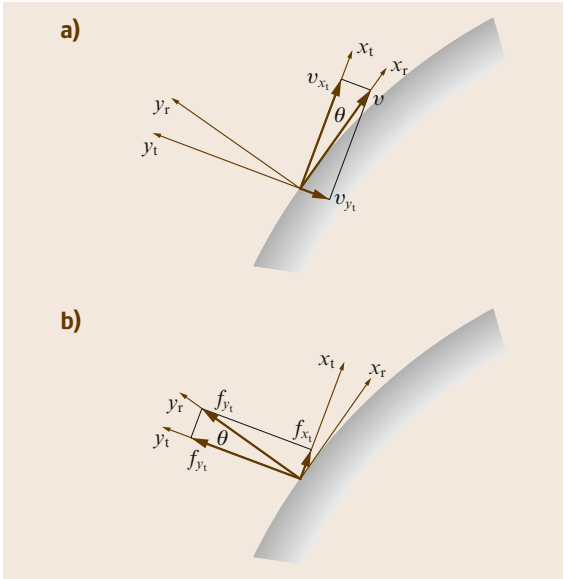


Fig.9.6a,b Estimation of orientation error: **(a)** velocity-based approach, **(b)** force-based approach

of the feature frames. The total model and the total set of constraints are achieved by collecting the partial task and constraints descriptions, each expressed in the individual feature frames.

9.3.4 Sensor-Based Contact Model Estimation

The task specification relies on the definition of the velocity-controlled subspace and of the force-controlled subspace assuming that an accurate model of the contact is available all the time. On the other hand, in most practical implementations, the selection matrices $\mathbf{S}_f$ and $\mathbf{S}_v$ are not exactly known, however many interaction control strategies turn out to be rather robust against modeling errors. In fact, to cope reliably with these situations is exactly why force control is used. The robustness of the force controller increases if the matrices $\mathbf{S}_f$ and $\mathbf{S}_v$ can be continuously updated, using motion and/or force measurements, during task execution.

In detail, a nominal model is assumed to be available; when the contact situation evolves differently from what the model predicts, the measured and predicted motion and force will begin to deviate. These small incompatibilities can be measured and can then be used to adapt the model online, using algorithms derived from classical state-space *prediction–correction* estimators such as the Kalman filter.

Figure 9.6 reports an example of error between nominal and measured motion and force variables, typical of a two-dimensional contour-following task. The orientation of the contact normal changes if the environment is notplanar. Hence an angular error $\boldsymbol{\theta}$ appears between the nominal contact normal, aligned to the y_t-axis of the task frame (the frame with axes x_t and y_t), and the real contact normal, aligned to the y_r-axis of the real task frame (the frame with axes x_r and y_r). This angle can be estimated with either velocity or force measurements only:

- *Velocity-based approach*: the executed linear velocity $\boldsymbol{v}$, which is tangent to the real contour (aligned to the x_r-axis), does not completely lie along the x_t-axis, but has a small component v_{y_t} along the y_t-axis. The orientation error $\boldsymbol{\theta}$ can then be approximated by $\boldsymbol{\theta} = \tan^{-1}(v_{y_t}/v_{x_t})$.
- *Force-based approach*: the measured (ideal) contact force $\boldsymbol{f}$ does not completely lie along the nominal normal direction, aligned to the y_t-axis, but has a small component f_{x_t} along the x_t-axis. The orientation error $\boldsymbol{\theta}$ can then be approximated by $\boldsymbol{\theta} = \tan^{-1}(f_{x_t}/f_{y_t})$.

The velocity-based approach is disturbed by mechanical compliance in the system; the force-based approach is disturbed by contact friction.

9.4 Hybrid Force/Motion Control

The aim of hybrid force/motion control is to split up simultaneous control of both end-effector motion and contact forces into two separate decoupled subproblems. In the following, the main control approaches in the hybrid framework are presented for the cases of both rigid and compliant environments.

9.4.1 Acceleration-Resolved Approach

As for the case of motion control, the acceleration-resolved approach is aimed at decoupling and linearizing the nonlinear robot dynamics at the acceleration level, via an inverse dynamics control law. In the presence of interaction with the environment, a complete decoupling between the force- and velocity-controlled subspaces is sought. The basic idea is that of designing a model-based *inner control loop* to compensate for the nonlinear dynamics of the robot manipulator and decouple the force and velocity subspaces; hence an *outer control loop* is designed to ensure disturbance rejection and tracking of the desired end-effector force and motion.

Avidbots Ex1016
Page 256 of 2154

Rigid Environment

In the case of a rigid environment, the external wrench can be written in the form $\boldsymbol{h}_e = \mathbf{S}_f \boldsymbol{\lambda}$. The force multiplier $\boldsymbol{\lambda}$ can be computed from (9.2) by multiplying both sides of (9.2) by the weighted pseudo-inverse $\mathbf{S}_f^{\dagger}$ with weight $\boldsymbol{\Lambda}^{-1}(\boldsymbol{q})$ and using the time derivative of the last equality (9.35). This yields

$$\boldsymbol{\lambda} = \mathbf{S}_f^{\dagger}(\boldsymbol{q})\,[\boldsymbol{h}_c - \boldsymbol{\mu}(\boldsymbol{q}, \dot{\boldsymbol{q}})] + \boldsymbol{\Lambda}_f(\boldsymbol{q})\dot{\mathbf{S}}_f^{\mathrm{T}} \boldsymbol{v}_e \,, \tag{9.51}$$

where $\boldsymbol{\Lambda}_f(\boldsymbol{q}) = (\mathbf{S}_f^{\mathrm{T}} \boldsymbol{\Lambda}^{-1} \mathbf{S}_f)^{-1}$ and $\boldsymbol{\mu}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \boldsymbol{\Gamma}\dot{\boldsymbol{q}} + \boldsymbol{\eta}$. Replacing again (9.51) into (9.2) gives

$$\boldsymbol{\Lambda}(\boldsymbol{q})\dot{\boldsymbol{v}}_e + \mathbf{S}_f \boldsymbol{\Lambda}_f(\boldsymbol{q})\dot{\mathbf{S}}_f^{\mathrm{T}} \boldsymbol{v}_e = \mathbf{N}_f(\boldsymbol{q})[\boldsymbol{h}_c - \boldsymbol{\mu}(\boldsymbol{q}, \dot{\boldsymbol{q}})] \,, \tag{9.52}$$

with $\mathbf{N}_f = \mathbf{I} - \mathbf{S}_f \mathbf{S}_f^{\dagger}$. Notice that $\mathbf{N}_f \mathbf{N}_f = \mathbf{N}_f$ and $\mathbf{N}_f \mathbf{S}_f = \mathbf{0}$, hence the 6×6 matrix $\mathbf{N}_f$ is a projection matrix that filters out all the end-effector wrenches lying in the range of $\mathbf{S}_f$. These correspond to wrenches that tend to violate the constraints. Equation (9.52) represents a set of six second-order differential equations whose solution, if initialized on the constraints, automatically satisfy (9.29) at all times.

The reduced-order dynamics of the constrained system is described by $6 - m$ second-order equations that are obtained by premultiplying both sides of (9.52) by the matrix $\mathbf{S}_v^{\mathrm{T}}$ and substituting the acceleration $\dot{\boldsymbol{v}}_e$ with

$$\dot{\boldsymbol{v}}_e = \mathbf{S}_v \dot{\boldsymbol{\nu}} + \dot{\mathbf{S}}_v \boldsymbol{\nu} \,.$$

The resulting equations are

$$\boldsymbol{\Lambda}_v(\boldsymbol{q})\dot{\boldsymbol{\nu}} = \mathbf{S}_v^{\mathrm{T}} \left[\boldsymbol{h}_c - \boldsymbol{\mu}(\boldsymbol{q}, \dot{\boldsymbol{q}}) - \boldsymbol{\Lambda}(\boldsymbol{q})\dot{\mathbf{S}}_v \boldsymbol{\nu} \right] \,, \tag{9.53}$$

where

$$\boldsymbol{\Lambda}_v = \mathbf{S}_v^{\mathrm{T}} \boldsymbol{\Lambda} \mathbf{S}_v$$

and the identities (9.38) and

$$\mathbf{S}_v^{\mathrm{T}} \mathbf{N}_f = \mathbf{S}_v^{\mathrm{T}}$$

have been used. Moreover, expression (9.51) can be rewritten as

$$\boldsymbol{\lambda} = \mathbf{S}_f^{\dagger}(\boldsymbol{q}) \left[\boldsymbol{h}_c - \boldsymbol{\mu}(\boldsymbol{q}, \dot{\boldsymbol{q}}) - \boldsymbol{\Lambda}(\boldsymbol{q})\dot{\mathbf{S}}_v \boldsymbol{\nu} \right] \,, \tag{9.54}$$

where the identity $\dot{\mathbf{S}}_f^{\mathrm{T}} \mathbf{S}_v = -\mathbf{S}_f^{\mathrm{T}} \dot{\mathbf{S}}_v$ has been exploited.

Equation (9.54) reveals that the force multiplier vector $\boldsymbol{\lambda}$ instantaneously depends also on the applied input wrench $\boldsymbol{h}_c$. Hence, by suitably choosing $\boldsymbol{h}_c$, it is possible to directly control the m independent components of the end-effector wrench that tend to violate the constraints; these components can be computed from the m force multipliers through (9.31). On the other hand, the $6 - m$ independent components of the end-effector velocity twist can be controlled through $\boldsymbol{h}_c$ via (9.53).

An inverse-dynamics *inner control loop* can be designed by choosing the control wrench $\boldsymbol{h}_c$ as

$$\boldsymbol{h}_c = \boldsymbol{\Lambda}(\boldsymbol{q})\mathbf{S}_v \boldsymbol{\alpha}_v + \mathbf{S}_f \boldsymbol{f}_\lambda + \boldsymbol{\mu}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \boldsymbol{\Lambda}(\boldsymbol{q})\dot{\mathbf{S}}_v \boldsymbol{\nu} \,, \tag{9.55}$$

where $\boldsymbol{\alpha}_v$ and $\boldsymbol{f}_\lambda$ are properly designed control inputs.

Substituting (9.55) into (9.53) and (9.51) yields

$$\dot{\boldsymbol{\nu}} = \boldsymbol{\alpha}_v \,,$$
$$\boldsymbol{\lambda} = \boldsymbol{f}_\lambda \,,$$

showing that control law (9.55) allows complete decoupling between the force- and velocity-controlled subspaces.

It is worth noticing that, for the implementation of the control law (9.55), the constraint (9.29) as well as (9.40) defining the vector of the configuration variables for the constrained system are not required, provided that the matrices $\mathbf{S}_f$ and $\mathbf{S}_v$ are known or estimated online. In these cases, the task can easily be assigned by specifying a desired force, in terms of the vector $\boldsymbol{\lambda}_d(t)$, and a desired velocity, in terms of the vector $\boldsymbol{\nu}_d(t)$; moreover, a force/velocity control is implemented.

The desired force $\boldsymbol{\lambda}_d(t)$ can be achieved by setting

$$\boldsymbol{f}_\lambda = \boldsymbol{\lambda}_d(t) \,, \tag{9.56}$$

but this choice is very sensitive to disturbance forces, since it contains no force feedback. Alternative choices are

$$\boldsymbol{f}_\lambda = \boldsymbol{\lambda}_d(t) + \mathbf{K}_{P\lambda} [\boldsymbol{\lambda}_d(t) - \boldsymbol{\lambda}(t)] \,, \tag{9.57}$$

or

$$\boldsymbol{f}_\lambda = \boldsymbol{\lambda}_d(t) + \mathbf{K}_{I\lambda} \int_0^t [\boldsymbol{\lambda}_d(\tau) - \boldsymbol{\lambda}(\tau)] \, \mathrm{d}\tau \,, \tag{9.58}$$

where $\mathbf{K}_{P\lambda}$ and $\mathbf{K}_{I\lambda}$ are suitable positive-definite matrix gains. The proportional feedback is able to reduce the force error due to disturbance forces, while the integral action is able to compensate for constant bias disturbances.

The implementation of force feedback requires the computation of the force multiplier λ from the measurement of the end-effector wrench $\boldsymbol{h}_e$, which can be achieved by using (9.33).

Velocity control is achieved by setting

$$\boldsymbol{\alpha}_v = \dot{\boldsymbol{\nu}}_d(t) + \mathbf{K}_{P\nu} [\boldsymbol{\nu}_d(t) - \boldsymbol{\nu}(t)] + \mathbf{K}_{I\nu} \int_0^t [\boldsymbol{\nu}_d(\tau) - \boldsymbol{\nu}(\tau)] \, \mathrm{d}\tau \,, \tag{9.59}$$

Avidbots Ex1016
Page 257 of 2154

where $\mathbf{K}_{P\nu}$ and $\mathbf{K}_{I\nu}$ are suitable matrix gains. It is straightforward to show that asymptotic tracking of $\boldsymbol{\nu}_d(t)$ and $\dot{\boldsymbol{\nu}}_d(t)$ is ensured with exponential convergence for any choice of positive-definite matrices $\mathbf{K}_{P\nu}$ and $\mathbf{K}_{I\nu}$.

The computation of the vector $\boldsymbol{\nu}$ from the available measurements can be achieved using (9.39), where the end-effector twist is computed from joint position and velocity measurements through (9.1).

Equations (9.57) or (9.58) and (9.59) represent the *outer control loop* ensuring *force/velocity* control and disturbance rejection.

When (9.29) and (9.40) are known, the matrices $\mathbf{S}_f$ and $\mathbf{S}_v$ can be computed according to (9.32) and (9.42) and a *force/position* control can be designed by specifying a desired force $\boldsymbol{\lambda}_d(t)$ and a desired position $\boldsymbol{r}_d(t)$.

Force control can be designed as above, while position control can be achieved by setting

$$\boldsymbol{\alpha}_\nu = \ddot{\boldsymbol{r}}_d(t) + \mathbf{K}_{Dr}\left[\dot{\boldsymbol{r}}_d(t) - \boldsymbol{\nu}(t)\right] + \mathbf{K}_{Pr}\left[\boldsymbol{r}_d(t) - \boldsymbol{r}(t)\right] .$$

Asymptotic tracking of $\boldsymbol{r}_d(t)$, $\dot{\boldsymbol{r}}_d(t)$ and $\ddot{\boldsymbol{r}}_d(t)$ is ensured with exponential convergence for any choice of positive-definite matrices $\mathbf{K}_{Dr}$ and $\mathbf{K}_{Pr}$. The vector $\boldsymbol{r}$ required for position feedback can be computed from joint position measurements via (9.40).

Compliant Environment

In the case of a compliant environment, according to the decomposition (9.44) of the end-effector displacement, the end-effector twist can be decomposed as

$$\boldsymbol{v}_e = \mathbf{S}_v\boldsymbol{\nu} + \mathbf{C}'\mathbf{S}_f\dot{\boldsymbol{\lambda}} , \tag{9.60}$$

where the first term is a twist of freedom, the second term is a twist of constraint, the vector $\boldsymbol{\nu}$ is defined as in (9.42), and $\mathbf{C}'$ is defined in (9.50). Assuming a constant contact geometry and compliance, i. e., $\dot{\mathbf{S}}_v = \mathbf{0}$, $\dot{\mathbf{C}}' = \mathbf{0}$, and $\dot{\mathbf{S}}_f = \mathbf{0}$, a similar decomposition holds in terms of acceleration

$$\dot{\boldsymbol{v}}_e = \mathbf{S}_v\dot{\boldsymbol{\nu}} + \mathbf{C}'\mathbf{S}_f\ddot{\boldsymbol{\lambda}} . \tag{9.61}$$

An inverse-dynamics control law (9.16) can be adopted, resulting in the closed loop (9.17), where $\boldsymbol{\alpha}$ is a properly designed control input.

In view of the acceleration decomposition (9.61), the choice

$$\boldsymbol{\alpha} = \mathbf{S}_v\boldsymbol{\alpha}_\nu + \mathbf{C}'\mathbf{S}_f\boldsymbol{f}_\lambda \tag{9.62}$$

allows decoupling of the force control from velocity control. In fact, substituting (9.61) and (9.62) into (9.17) and premultiplying both sides of the resulting equation once by $\mathbf{S}_v^\dagger$ and once by $\mathbf{S}_f^T$, the following decoupled equations can be derived

$$\dot{\boldsymbol{\nu}} = \boldsymbol{\alpha}_\nu , \tag{9.63}$$

$$\ddot{\boldsymbol{\lambda}} = \boldsymbol{f}_\lambda . \tag{9.64}$$

Hence, by choosing $\boldsymbol{\alpha}_\nu$ according to (9.59) as for the rigid environment case, asymptotic tracking of a desired velocity $\boldsymbol{\nu}_d(t)$ and acceleration $\dot{\boldsymbol{\nu}}_d(t)$ is ensured, with exponential convergence. The control input $\boldsymbol{f}_\lambda$ can be chosen as

$$\begin{aligned}\boldsymbol{f}_\lambda = {} & \ddot{\boldsymbol{\lambda}}_d(t) + \mathbf{K}_{D\lambda}\left[\dot{\boldsymbol{\lambda}}_d(t) - \dot{\boldsymbol{\lambda}}(t)\right] \\ & + \mathbf{K}_{P\lambda}\left[\boldsymbol{\lambda}_d(t) - \boldsymbol{\lambda}(t)\right] ,\end{aligned} \tag{9.65}$$

ensuring asymptotic tracking of a desired force trajectory $(\boldsymbol{\lambda}_d(t), \dot{\boldsymbol{\lambda}}_d(t), \ddot{\boldsymbol{\lambda}}_d(t))$ with exponential convergence for any choice of positive-definite matrices $\mathbf{K}_{D\lambda}$ and $\mathbf{K}_{P\lambda}$.

Differently from the rigid environment case, feedback of $\dot{\boldsymbol{\lambda}}$ is required for the implementation of the force control law (9.65). This quantity could be computed from end-effector wrench measurements $\boldsymbol{h}_e$ as

$$\dot{\boldsymbol{\lambda}} = \mathbf{S}_f^\dagger\dot{\boldsymbol{h}}_e .$$

However, since the wrench measurement signal is often noisy, the feedback of $\dot{\boldsymbol{\lambda}}$ is often replaced by

$$\dot{\boldsymbol{\lambda}} = \mathbf{S}_f^\dagger\mathbf{K}'\mathbf{J}(\boldsymbol{q})\dot{\boldsymbol{q}} , \tag{9.66}$$

where joint velocities are measured using tachometers or computed from joint positions via numerical differentiation and $\mathbf{K}'$ is the positive-semidefinite stiffness matrix (9.49) describing the partially constrained interaction. For the computation of (9.66), only knowledge (or an estimate) of $\mathbf{K}'$ is required, and not that of the full stiffness matrix $\mathbf{K}$. Also, the implementation of the control law (9.62) requires knowledge (or an estimate) of the compliance matrix $\mathbf{C}'$ of the partially constrained interaction and not that of the full compliance matrix $\mathbf{C}$.

If the contact geometry is known, but only an estimate of the stiffness/compliance of the environment is available, the control law (9.62), with (9.65), may still guarantee the convergence of the force error, if a constant desired force $\boldsymbol{\lambda}_d$ is assigned. In this case, the control law (9.62) has the form

$$\boldsymbol{\alpha} = \mathbf{S}_v\boldsymbol{\alpha}_\nu + \widehat{\mathbf{C}}'\mathbf{S}_f\boldsymbol{f}_\lambda ,$$

where $\widehat{\mathbf{C}}' = (\mathbf{I} - \mathbf{P}_v)\widehat{\mathbf{C}}$ and $\widehat{\mathbf{C}}$ is an estimate of the compliance matrix. Hence, (9.63) still holds, while, in lieu of (9.64), the following equality can be found

$$\ddot{\boldsymbol{\lambda}} = \mathbf{L}_f\boldsymbol{f}_\lambda ,$$

Avidbots Ex1016
Page 258 of 2154

where $\mathbf{L}_f = (\mathbf{S}_f^T \mathbf{C} \mathbf{S}_f)^{-1} \mathbf{S}_f^T \widehat{\mathbf{C}} \mathbf{S}_f$ is a nonsingular matrix. Thus, the force- and velocity-controlled subspaces remain decoupled and the velocity control law (9.59) does not need to be modified. On the other hand, if the feedback of the time derivative of $\boldsymbol{\lambda}$ is computed using (9.66), only an estimate $\dot{\hat{\boldsymbol{\lambda}}}$ can be obtained. Using (9.66), (9.60) and (9.48), the following equality can be found

$$\dot{\hat{\boldsymbol{\lambda}}} = \mathbf{L}_f^{-1} \dot{\boldsymbol{\lambda}} \,.$$

Therefore, computing the force control law $\boldsymbol{f}_\lambda$ as in (9.65) with a constant $\boldsymbol{\lambda}_d$, $\dot{\hat{\boldsymbol{\lambda}}}$ in lieu of $\dot{\boldsymbol{\lambda}}$ and $\mathbf{K}_{D\lambda} = \mathbf{K}_{D\lambda} \mathbf{I}$, the dynamics of the closed-loop system is

$$\ddot{\boldsymbol{\lambda}} + \mathbf{K}_{D\lambda} \dot{\boldsymbol{\lambda}} + \mathbf{L}_f \mathbf{K}_{P\lambda} \boldsymbol{\lambda} = \mathbf{L}_f \mathbf{K}_{P\lambda} \boldsymbol{\lambda}_d \,,$$

showing that exponential asymptotic stability of the equilibrium $\boldsymbol{\lambda} = \boldsymbol{\lambda}_d$ can be ensured, also in the presence of the uncertain matrix $\mathbf{L}_f$, with a suitable choice of the gains $\mathbf{K}_{D\lambda}$ and $\mathbf{K}_{P\lambda}$.

9.4.2 Passivity-Based Approach

The passivity-based approach exploits the passivity properties of the dynamic model of the manipulator, which hold also for the constrained dynamic model (9.2). It can be easily shown that the choice of the matrix $\mathbf{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ that guarantees the skew symmetry of the matrix $\dot{\mathbf{H}}(\boldsymbol{q}) - 2\mathbf{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ in the joint space, also makes the matrix $\dot{\boldsymbol{\Lambda}}(\boldsymbol{q}) - 2\boldsymbol{\Gamma}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ skew symmetric. This fundamental property of Lagrangian systems is at the base of passivity-based control algorithms.

Rigid Environment

The control wrench $\boldsymbol{h}_c$ can be chosen as

$$\boldsymbol{h}_c = \boldsymbol{\Lambda}(\boldsymbol{q}) \mathbf{S}_v \dot{\boldsymbol{\nu}}_r + \boldsymbol{\Gamma}'(\boldsymbol{q}, \dot{\boldsymbol{q}}) \boldsymbol{\nu}_r + (\mathbf{S}_v^\dagger)^T \mathbf{K}_\nu (\boldsymbol{\nu}_r - \boldsymbol{\nu}) + \boldsymbol{\eta}(\boldsymbol{q}) + \mathbf{S}_f \boldsymbol{f}_\lambda \,, \tag{9.67}$$

where $\boldsymbol{\Gamma}'(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \boldsymbol{\Gamma}\, \mathbf{S}_v + \boldsymbol{\Lambda} \dot{\mathbf{S}}_v$, $\mathbf{K}_\nu$ is a suitable symmetric and positive-definite matrix, and $\boldsymbol{\nu}_r$ and $\boldsymbol{f}_\lambda$ are properly designed control inputs.

Substituting (9.67) into (9.2) yields

$$\boldsymbol{\Lambda}(\boldsymbol{q}) \mathbf{S}_v \dot{\boldsymbol{s}}_\nu + \boldsymbol{\Gamma}'(\boldsymbol{q}, \dot{\boldsymbol{q}}) \boldsymbol{s}_\nu + (\mathbf{S}_v^\dagger)^T \mathbf{K}_\nu \boldsymbol{s}_\nu + \mathbf{S}_f (\boldsymbol{f}_\lambda - \boldsymbol{\lambda}) = \mathbf{0} \,, \tag{9.68}$$

with $\dot{\boldsymbol{s}}_\nu = \dot{\boldsymbol{\nu}}_r - \dot{\boldsymbol{\nu}}$ and $\boldsymbol{s}_\nu = \boldsymbol{\nu}_r - \boldsymbol{\nu}$, showing that the closed-loop system remains nonlinear and coupled.

Premultiplying both sides of (9.68) by the matrix $\mathbf{S}_v$, the following expression for the reduced-order dynamics is achieved

$$\boldsymbol{\Lambda}_v(\boldsymbol{q}) \dot{\boldsymbol{s}}_\nu + \boldsymbol{\Gamma}_v(\boldsymbol{q}, \dot{\boldsymbol{q}}) \boldsymbol{s}_\nu + \mathbf{K}_\nu \boldsymbol{s}_\nu = \mathbf{0} \,, \tag{9.69}$$

with $\boldsymbol{\Gamma}_v = \mathbf{S}_v^T \boldsymbol{\Gamma}(\boldsymbol{q}, \dot{\boldsymbol{q}}) \mathbf{S}_v + \mathbf{S}_v^T \boldsymbol{\Lambda}(\boldsymbol{q}) \dot{\mathbf{S}}_v$; it can easily be shown that the skew symmetry of the matrix $\dot{\boldsymbol{\Lambda}}(\boldsymbol{q}) - 2\boldsymbol{\Gamma}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ implies that the matrix $\dot{\boldsymbol{\Lambda}}_v(\boldsymbol{q}) - 2\boldsymbol{\Gamma}_v(\boldsymbol{q}, \dot{\boldsymbol{q}})$ is skew symmetric as well.

On the other hand, premultiplying both sides of (9.68) by the matrix $\mathbf{S}_f^T \boldsymbol{\Lambda}^{-1}(\boldsymbol{q})$, the following expression for the force dynamics can be found

$$\boldsymbol{f}_\lambda - \boldsymbol{\lambda} = -\mathbf{S}_f^\dagger(\boldsymbol{q}) \left[\boldsymbol{\Gamma}'(\boldsymbol{q}, \dot{\boldsymbol{q}}) - (\mathbf{S}_v^\dagger)^T \mathbf{K}_\nu \right] \boldsymbol{s}_\nu \,, \tag{9.70}$$

being $\mathbf{S}_f^\dagger$ the weighted pseudo-inverse of $\mathbf{S}_f$ with weight $\boldsymbol{\Lambda}^{-1}(\boldsymbol{q})$. The above equation shows that the force multiplier $\boldsymbol{\lambda}$ instantaneously depends on the control input $\boldsymbol{f}_\lambda$ but also on the error $\boldsymbol{s}_\nu$ in the velocity-controlled subspace.

The asymptotic stability of the reduced-order system (9.69) can be ensured with the choices

$$\dot{\boldsymbol{\nu}}_r = \dot{\boldsymbol{\nu}}_d + \boldsymbol{\alpha} \Delta \boldsymbol{\nu} \,, \tag{9.71}$$

$$\boldsymbol{\nu}_r = \boldsymbol{\nu}_d + \boldsymbol{\alpha} \Delta \boldsymbol{x}_\nu \,, \tag{9.72}$$

where $\boldsymbol{\alpha}$ is a positive gain, $\dot{\boldsymbol{\nu}}_d$ and $\boldsymbol{\nu}_d$ are the desired acceleration and velocity, respectively, $\Delta \boldsymbol{\nu} = \boldsymbol{\nu}_d - \boldsymbol{\nu}$, and

$$\Delta \boldsymbol{x}_\nu = \int_0^t \Delta \boldsymbol{\nu}(\tau) \, \mathrm{d}\tau \,.$$

The stability proof is based on the positive-definite Lyapunov function

$$V = \frac{1}{2} \boldsymbol{s}_\nu^T \boldsymbol{\Lambda}_v(\boldsymbol{q}) \boldsymbol{s}_\nu + \boldsymbol{\alpha} \Delta \boldsymbol{x}_\nu^T \mathbf{K}_\nu \Delta \boldsymbol{x}_\nu \,,$$

whose time derivative along the trajectories of (9.69),

$$\dot{V} = -\Delta \boldsymbol{\nu}^T \mathbf{K}_\nu \Delta \boldsymbol{\nu} - \boldsymbol{\alpha}^2 \Delta \boldsymbol{x}_\nu^T \mathbf{K}_\nu \Delta \boldsymbol{x}_\nu \,,$$

is a definite semi-negative function. Hence, $\Delta \boldsymbol{\nu} = \mathbf{0}$, $\Delta \boldsymbol{x}_\nu = \mathbf{0}$, and $\boldsymbol{s}_\nu = \mathbf{0}$ asymptotically. Therefore, tracking of the desired velocity $\boldsymbol{\nu}_d(t)$ is ensured. Moreover, the right-hand side of (9.70) remains bounded and vanishes asymptotically. Hence, tracking of the desired force $\boldsymbol{\lambda}_d(t)$ can be ensured by setting $\boldsymbol{f}_\lambda$ as for the acceleration-resolved approach, according to the choices (9.56)–(9.58).

Notice that position control can be achieved if a desired position $\boldsymbol{r}_d(t)$ is assigned for the vector $\boldsymbol{r}$ in (9.40), provided that the matrices $\mathbf{S}_f$ and $\mathbf{S}_v$ are computed according to (9.32) and (9.42), and the vectors $\dot{\boldsymbol{\nu}}_d = \ddot{\boldsymbol{r}}_d$, $\boldsymbol{\nu}_d = \dot{\boldsymbol{r}}_d$, and $\Delta \boldsymbol{x}_\nu = \boldsymbol{r}_d - \boldsymbol{r}$ are used in (9.71) and (9.72).

Avidbots Ex1016
Page 259 of 2154

Compliant Environment

The control wrench $\boldsymbol{h}_c$ can be chosen as

$$\boldsymbol{h}_c = \boldsymbol{\Lambda}(\boldsymbol{q})\dot{\boldsymbol{v}}_r + \boldsymbol{\Gamma}(\boldsymbol{q},\dot{\boldsymbol{q}})\boldsymbol{v}_r + \mathbf{K}_s(\boldsymbol{v}_r - \boldsymbol{v}_e) + \boldsymbol{h}_e + \boldsymbol{\eta}(\boldsymbol{q}) , \tag{9.73}$$

where $\mathbf{K}_s$ is a suitable symmetric positive-definite matrix while $\boldsymbol{v}_r$ and its time derivative $\dot{\boldsymbol{v}}_r$ are chosen as

$$\boldsymbol{v}_r = \boldsymbol{v}_d + \boldsymbol{\alpha}\Delta\boldsymbol{x} ,$$
$$\dot{\boldsymbol{v}}_r = \dot{\boldsymbol{v}}_d + \boldsymbol{\alpha}\Delta\boldsymbol{v} ,$$

where $\boldsymbol{\alpha}$ is the positive gain, $\boldsymbol{v}_d$ and its time derivative $\dot{\boldsymbol{v}}_d$ are properly designed control inputs, $\Delta\boldsymbol{v} = \boldsymbol{v}_d - \boldsymbol{v}_e$, and

$$\Delta\boldsymbol{x} = \int_0^t \Delta\boldsymbol{v}\,\mathrm{d}\boldsymbol{\tau} .$$

Substituting (9.73) into (9.2) yields

$$\boldsymbol{\Lambda}(\boldsymbol{q})\dot{\boldsymbol{s}} + \boldsymbol{\Gamma}(\boldsymbol{q},\dot{\boldsymbol{q}})\boldsymbol{s} + \mathbf{K}_s\boldsymbol{s} = \mathbf{0} , \tag{9.74}$$

with $\dot{\boldsymbol{s}} = \dot{\boldsymbol{v}}_r - \dot{\boldsymbol{v}}_e$ and $\boldsymbol{s} = \boldsymbol{v}_r - \boldsymbol{v}_e$.

The asymptotic stability of system (9.74) can be ensured by setting

$$\boldsymbol{v}_d = \mathbf{S}_v\boldsymbol{\nu}_d + \mathbf{C}'\mathbf{S}_f\dot{\boldsymbol{\lambda}}_d ,$$

where $\boldsymbol{\nu}_d(t)$ is a desired velocity trajectory and $\boldsymbol{\lambda}_d(t)$ is the desired force trajectory. The stability proof is based on the positive-definite Lyapunov function

$$V = \frac{1}{2}\boldsymbol{s}^{\mathrm{T}}\boldsymbol{\Lambda}(\boldsymbol{q})\boldsymbol{s} + \boldsymbol{\alpha}\Delta\boldsymbol{x}^{\mathrm{T}}\mathbf{K}_s\Delta\boldsymbol{x} ,$$

whose time derivative along the trajectories of (9.74),

$$\dot{V} = -\Delta\boldsymbol{v}^{\mathrm{T}}\mathbf{K}_s\Delta\boldsymbol{v} - \boldsymbol{\alpha}^2\Delta\boldsymbol{x}^{\mathrm{T}}\mathbf{K}_s\Delta\boldsymbol{x} ,$$

is a negative-definite function. Hence, $\Delta\boldsymbol{v} = \mathbf{0}$ and $\Delta\boldsymbol{x} = \mathbf{0}$, asymptotically. In the case of constant contact geometry and stiffness, the following equalities hold

$$\Delta\boldsymbol{v} = \mathbf{S}_v(\boldsymbol{\nu}_d - \boldsymbol{\nu}) + \mathbf{C}'\mathbf{S}_f(\dot{\boldsymbol{\lambda}}_d - \dot{\boldsymbol{\lambda}}) ,$$
$$\Delta\boldsymbol{x} = \mathbf{S}_v\int_0^t (\boldsymbol{\nu}_d - \boldsymbol{\nu})\,\mathrm{d}\boldsymbol{\tau} + \mathbf{C}'\mathbf{S}_f(\boldsymbol{\lambda}_d - \boldsymbol{\lambda}) ,$$

showing that both the velocity and force tracking errors, belonging to reciprocal subspaces, converge asymptotically to zero.

9.4.3 Velocity-Resolved Approach

The acceleration-resolved approach, as well as the passivity-based approach, require modification of the current industrial robot controllers. As for the case of impedance control, if the contact is sufficiently compliant, the closed-loop dynamics of a motion-controlled robot can be approximated by (9.28), corresponding to a velocity-resolved control.

To achieve force and velocity control, according to the end-effector twist decomposition (9.60), the control input $\boldsymbol{v}_r$ can be chosen as

$$\boldsymbol{v}_r = \mathbf{S}_v\boldsymbol{v}_\nu + \mathbf{C}'\mathbf{S}_f\boldsymbol{f}_\lambda , \tag{9.75}$$

with

$$\boldsymbol{v}_\nu = \boldsymbol{\nu}_d(t) + \mathbf{K}_{I\nu}\int_0^t [\boldsymbol{\nu}_d(\boldsymbol{\tau}) - \boldsymbol{\nu}(\boldsymbol{\tau})]\,\mathrm{d}\boldsymbol{\tau} , \tag{9.76}$$

and

$$\boldsymbol{f}_\lambda = \dot{\boldsymbol{\lambda}}_d(t) + \mathbf{K}_{P\lambda}[\boldsymbol{\lambda}_d(t) - \boldsymbol{\lambda}(t)] , \tag{9.77}$$

where $\mathbf{K}_{I\nu}$ and $\mathbf{K}_{P\lambda}$ are suitable symmetric and positive-definite matrix gains. Decoupling between velocity- and force-controlled subspaces and exponential asymptotic stability of the closed-loop system can be proven as for the acceleration-resolved approach. Also, since the force error has second-order dynamics, an integral action can be added to (9.77) to improve the disturbance rejection capabilities, i. e.,

$$\boldsymbol{f}_\lambda = \dot{\boldsymbol{\lambda}}_d(t) + \mathbf{K}_{P\lambda}[\boldsymbol{\lambda}_d(t) - \boldsymbol{\lambda}(t)] + \mathbf{K}_{I\lambda}\int_0^t [\boldsymbol{\lambda}_d(\boldsymbol{\tau}) - \boldsymbol{\lambda}(\boldsymbol{\tau})]\,\mathrm{d}\boldsymbol{\tau} , \tag{9.78}$$

and the exponential asymptotic stability is guaranteed if the matrices $\mathbf{K}_{P\lambda}$ and $\mathbf{K}_{I\lambda}$ are symmetric and positive definite.

If an estimate $\widehat{\mathbf{C}}$ of the stiffness matrix of the environment is used in (9.75), as for the acceleration-resolved approach, the exponential convergence of $\boldsymbol{\lambda}$ to a constant $\boldsymbol{\lambda}_d$ can still be ensured for both (9.77) and (9.78).

In some applications, besides the stiffness matrix, also the geometry of the environment is uncertain. In these cases, a force/motion control law similar to (9.75) can be implemented, without using the selection matrices $\mathbf{S}_v$ and $\mathbf{S}_f$ to separate the force-controlled subspace from the velocity-controlled subspace. The motion control law can be set as in (9.76),

Avidbots Ex1016
Page 260 of 2154

but using full velocity feedback. Also, the force control law can be set as in (9.78), but using full force and moment feedback. That is, both motion control and force control are applied in all directions of the six-dimensional (6-D) space. The resulting control, known as *force control with feedforward motion* or *parallel force/position control* guarantees force regulation at the expense of position errors along the constrained task directions, thanks to the dominance of the force controller over the position controller ensured by the presence of the integral action on the force error. This approach is tested in the experiments reported in VIDEO 687 and VIDEO 691 where the concept of task frame is also exploited. A framework for robotic assembly, where a standard position-based robot controller is integrated with an external controller performing force-controlled skills is presented in VIDEO 692.

9.5 Conclusions and Further Reading

This chapter has summarized the main approaches to force control in a unifying perspective. However, there are many aspects that have not been considered and that must be carefully taken into account when dealing with interaction robotic tasks. The two major paradigms of force control (impedance and hybrid force/motion control) are based on several simplifying assumptions that are only partially satisfied in practical implementations. In fact, the performance of a force-controlled robotic system depends on the interaction with a changing environment which is very difficult to model and identify correctly. A general contact situation is far from being completely predictable, not only quantitatively, but also qualitatively: the contact configuration can change abruptly, or be of a different type than expected. Hence, the standard performance indices used to evaluate a control system, i. e., stability, bandwidth, accuracy, and robustness, cannot be defined by considering the robotic system alone, as for the case of robot motion control, but must be always referred to the particular contact situation at hand. Also, a classification of all these different situations is not easy, especially in the case of dynamics environments and when the task involves multiple contacts acting in parallel.

Due to the inherent complexity of the force control problem, a large number of research papers on this topic have been published in the past three decades. A description of the state of the art of the first decade is provided in [9.20], whereas the progress of the second decade is surveyed in [9.21] and [9.22]. More recently, two monographs on force control [9.23, 24] have appeared. In the following, a list of references is provided, where more details on the arguments presented in the chapter, as well as topics not covered here, can be found.

9.5.1 Indirect Force Control

The concept of generalized spring and damping for force control in joint coordinates was originally proposed in [9.3] and the implementation discussed in [9.10]. Stiffness control in Cartesian coordinates was proposed in [9.9]. Devices based on the remote center of compliance were discussed in [9.25] for successful mating of rigid parts. The original idea of a mechanical impedance model used for controlling the interaction between the manipulator and the environment is presented in [9.7], and a similar formulation is given in [9.8]. Stability of impedance control was analyzed in [9.26] and the problems of interaction with a stiff environment were considered in [9.27].

Adaptive impedance control algorithms [9.28, 29] have been proposed to overcome uncertainties in the dynamic parameters of the robot manipulator, while robust control schemes can be found in [9.30]. Impedance control has also been used in the hybrid force/motion control framework [9.31].

A reference work on modeling 6-DOF (spatial) stiffness is [9.32], while the properties of spatial compliance have been analyzed in detail in [9.33–35]; a 6-DOF variable compliant wrist was proposed in [9.36], while several studies concerning the construction of programmed compliances, optimized for specific tasks, have been proposed [9.37, 38]. The energy-based approach to derive a spatial impedance was introduced in [9.39], using rotation matrices; various 6-DOF impedance control schemes based on different representations of end-effector orientation, including the unit quaternion, can be found in [9.40]. The quaternion-based formulation is extended to the case of non-block-diagonal stiffness matrix in [9.41]. A rigorous treatment of spatial impedance control in a passivity framework can be found in [9.42].

More recently, impedance control was proposed as an effective approach to enhance safety in applications where humans and robots share the same workspace and may have physical interaction. To this end, the passive compliance of lightweight robots is combined with the active compliance ensured by impedance control [9.43, 44].

Avidbots Ex1016
Page 261 of 2154

9.5.2 Task Specification

The concepts of natural and artificial constraints and of compliance frame were introduced in [9.11]. These ideas have been systematically developed in [9.12, 45] within the *task frame formalism*. Theoretical issues on the reciprocity of generalized force and velocity directions are discussed in [9.46, 47], while invariance in computing the generalized inverse in robotics is addressed in [9.48]. The issue of partially constrained tasks is considered in [9.49], where the models of positive-semidefinite stiffness and compliance matrices are developed. The problem of estimating geometric uncertainties is considered in [9.50, 51], as well as the issue of linking constraint-based task specification with real-time task execution control. This approach is generalized in [9.52], where a systematic constraint-based methodology to specify complex tasks is presented.

9.5.3 Hybrid Force/Motion Control

Early works on force control can be found in [9.10]. The original hybrid force/positon control concept was introduced in [9.13], based on the natural and artificial constraint task formulation [9.11]. The explicit inclusion of the manipulator dynamic model was presented in [9.17], and a systematic approach to modeling the interaction with a dynamic environment was developed in [9.53]. The constrained formulation with inverse dynamic controllers is treated in [9.14, 54] in the Cartesian space as well as in [9.15] joint space. The constrained approach was also used in [9.16] with a controller based on linearized equations. The invariance problems pointed out in [9.47] were correctly addressed, among other papers, in [9.46, 55]. Transposition of model-based schemes from unconstrained motion control to constrained cases was accomplished for adaptive control in [9.18, 56, 57] and for robust control in [9.58].

Approaches designed to cope with uncertainties in the environment geometry are the force control with feedforward motion scheme proposed in [9.2] and the parallel force/position control [9.19], based on the concept of dominance of force control on motion control, thanks to the use of an integral action on the force error. A parallel force/position regulator was developed in [9.59]. The integral action for removing steady-state force errors has traditionally been used; its stability was proven in [9.60], while robustness with respect to force measurement delays was investigated in [9.61, 62].

In the absence of force/torque sensors, suitable algorithms can be adopted to estimate the contact force, based, e.g., on disturbance observers [9.63], or on the error of a position control [9.64].

It has generally been recognized that force control may cause unstable behavior during contact with environment. Dynamic models for explaining this phenomenon were introduced in [9.65] and experimental investigations can be found in [9.66] and [9.67]. Moreover, control schemes are usually derived on the assumption that the manipulator end-effector is in contact with the environment and that this contact is not lost. Impact phenomena may occur and deserve careful consideration, and there is a need for global analysis of control schemes including the transition from noncontact to contact situations and vice versa, see e.g., [9.68–70].

Video-References

VIDEO 684 Recent research in impedance control
available from http://handbookofrobotics.org/view-chapter/09/videodetails/684

VIDEO 685 Integration of force strategies and natural admittance control
available from http://handbookofrobotics.org/view-chapter/09/videodetails/685

VIDEO 686 Experiments of spatial impedance control
available from http://handbookofrobotics.org/view-chapter/09/videodetails/686

VIDEO 687 Compliant robot motion; Control and task specification
available from http://handbookofrobotics.org/view-chapter/09/videodetails/687

VIDEO 691 COMRADE: Compliant motion research and development environment
available from http://handbookofrobotics.org/view-chapter/09/videodetails/691

VIDEO 692 Robotic assembly of emergency stop buttons
available from http://handbookofrobotics.org/view-chapter/09/videodetails/692

Avidbots Ex1016
Page 262 of 2154

References

9.1 T.L. De Fazio, D.S. Seltzer, D.E. Whitney: The instrumented remote center of compliance, Ind. Robot **11**(4), 238–242 (1984)

9.2 J. De Schutter, H. Van Brussel: Compliant robot motion II. A control approach based on external control loops, Int. J. Robotics Res. **7**(4), 18–33 (1988)

9.3 I. Nevins, D.E. Whitney: The force vector assembler concept, Proc. 1 CISM-IFToMM Symp. Theory Pract. Robotics Manip., Udine (1973)

9.4 M.T. Mason, J.K. Salisbury: *Robot Hands and Mechanics of Manipulation* (MIT Press, Cambridge 1985)

9.5 J.Y.S. Luh, W.D. Fisher, R.P.C. Paul: Joint torque control by direct feedback for industrial robots, IEEE Trans. Autom. Control **28**, 153–161 (1983)

9.6 G. Hirzinger, N. Sporer, A. Albu-Schäffer, M. Hähnle, R. Krenn, A. Pascucci, R. Schedl: DLR's torque-controlled light weight robot III – Are we reaching the technological limits now?, Proc. IEEE Int. Conf. Robotics Autom., Washington (2002) pp. 1710–1716

9.7 N. Hogan: Impedance control: An approach to manipulation: Parts I–III, ASME J. Dyn. Syst. Meas. Control **107**, 1–24 (1985)

9.8 H. Kazerooni, T.B. Sheridan, P.K. Houpt: Robust compliant motion for manipulators. Part I: The fundamental concepts of compliant motion, IEEE J. Robotics Autom. **2**, 83–92 (1986)

9.9 J.K. Salisbury: Active stiffness control of a manipulator in Cartesian coordinates, 19th IEEE Conf. Decis. Control, Albuquerque (1980) pp. 95–100

9.10 D.E. Whitney: Force feedback control of manipulator fine motions, ASME J. Dyn. Syst. Meas. Control **99**, 91–97 (1977)

9.11 M.T. Mason: Compliance and force control for computer controlled manipulators, IEEE Trans. Syst. Man Cybern. **11**, 418–432 (1981)

9.12 J. De Schutter, H. Van Brussel: Compliant robot motion I. A formalism for specifying compliant motion tasks, Int. J. Robotics Res. **7**(4), 3–17 (1988)

9.13 M.H. Raibert, J.J. Craig: Hybrid position/force control of manipulators, ASME J. Dyn. Syst. Meas. Control **103**, 126–133 (1981)

9.14 T. Yoshikawa: Dynamic hybrid position/force control of robot manipulators – Description of hand constraints and calculation of joint driving force, IEEE J. Robotics Autom. **3**, 386–392 (1987)

9.15 N.H. McClamroch, D. Wang: Feedback stabilization and tracking of constrained robots, IEEE Trans. Autom. Control **33**, 419–426 (1988)

9.16 J.K. Mills, A.A. Goldenberg: Force and position control of manipulators during constrained motion tasks, IEEE Trans. Robotics Autom. **5**, 30–46 (1989)

9.17 O. Khatib: A unified approach for motion and force control of robot manipulators: The operational space formulation, IEEE J. Robotics Autom. **3**, 43–53 (1987)

9.18 L. Villani, C. Canudas de Wit, B. Brogliato: An exponentially stable adaptive control for force and position tracking of robot manipulators, IEEE Trans. Autom. Control **44**, 798–802 (1983)

9.19 S. Chiaverini, L. Sciavicco: The parallel approach to force/position control of robotic manipulators, IEEE Trans. Robotics Autom. **9**, 361–373 (1993)

9.20 D.E. Whitney: Historical perspective and state of the art in robot force control, Int. J. Robotics Res. **6**(1), 3–14 (1987)

9.21 M. Vukobratović, Y. Nakamura: Force and contact control in robotic systems, Proc. IEEE Int. Conf. Robotics Autom., Atlanta (1993)

9.22 J. De Schutter, H. Bruyninckx, W.H. Zhu, M.W. Spong: Force control: A bird's eye view. In: *Control Problems in Robotics and Automation*, ed. by K.P. Valavanis, B. Siciliano (Springer, London 1998) pp. 1–17

9.23 D.M. Gorinevski, A.M. Formalsky, A.Y. Schneider: *Force Control of Robotics Systems* (CRC, Boca Raton 1997)

9.24 B. Siciliano, L. Villani: *Robot Force Control* (Kluwer, Boston 1999)

9.25 D.E. Whitney: Quasi-static assembly of compliantly supported rigid parts, ASME J. Dyn. Syst. Meas. Control **104**, 65–77 (1982)

9.26 N. Hogan: On the stability of manipulators performing contact tasks, IEEE J. Robotics Autom. **4**, 677–686 (1988)

9.27 H. Kazerooni: Contact instability of the direct drive robot when constrained by a rigid environment, IEEE Trans. Autom. Control **35**, 710–714 (1990)

9.28 R. Kelly, R. Carelli, M. Amestegui, R. Ortega: Adaptive impedance control of robot manipulators, IASTED Int. J. Robotics Autom. **4**(3), 134–141 (1989)

9.29 R. Colbaugh, H. Seraji, K. Glass: Direct adaptive impedance control of robot manipulators, J. Robotics Syst. **10**, 217–248 (1993)

9.30 Z. Van Lu, A.A. Goldenberg: Robust impedance control and force regulation: Theory and experiments, Int. J. Robotics Res. **14**, 225–254 (1995)

9.31 R.J. Anderson, M.W. Spong: Hybrid impedance control of robotic manipulators, IEEE J. Robotics Autom. **4**, 549–556 (1986)

9.32 J. Lončarić: Normal forms of stiffness and compliance matrices, IEEE J. Robotics Autom. **3**, 567–572 (1987)

9.33 T. Patterson, H. Lipkin: Structure of robot compliance, ASME J. Mech. Design **115**, 576–580 (1993)

9.34 E.D. Fasse, P.C. Breedveld: Modelling of elastically coupled bodies: Part I – General theory and geometric potential function method, ASME J. Dyn. Syst. Meas. Control **120**, 496–500 (1998)

9.35 E.D. Fasse, P.C. Breedveld: Modelling of elastically coupled bodies: Part II – Exponential and generalized coordinate method, ASME J. Dyn. Syst. Meas. Control **120**, 501–506 (1998)

9.36 R.L. Hollis, S.E. Salcudean, A.P. Allan: A six-degree-of-freedom magnetically levitated variable compliance fine-motion wrist: Design, modeling and control, IEEE Trans. Robotics Autom. **7**, 320–333 (1991)

9.37 M.A. Peshkin: Programmed compliance for error corrective assembly, IEEE Trans. Robotics Autom. **6**, 473–482 (1990)

Avidbots Ex1016
Page 263 of 2154

9.38 J.M. Shimmels, M.A. Peshkin: Admittance matrix design for force-guided assembly, IEEE Trans. Robotics Autom. **8**, 213–227 (1992)

9.39 E.D. Fasse, J.F. Broenink: A spatial impedance controller for robotic manipulation, IEEE Trans. Robotics Autom. **13**, 546–556 (1997)

9.40 F. Caccavale, C. Natale, B. Siciliano, L. Villani: Six-DOF impedance control based on angle/axis representations, IEEE Trans. Robotics Autom. **15**, 289–300 (1999)

9.41 F. Caccavale, C. Natale, B. Siciliano, L. Villani: Robot impedance control with nondiagonal stiffness, IEEE Trans. Autom. Control **44**, 1943–1946 (1999)

9.42 S. Stramigioli: *Modeling and IPC Control of Interactive Mechanical Systems – A Coordinate Free Approach* (Springer, London 2001)

9.43 C. Ott: *Cartesian Impedance Control of Redundant and Flexible-Joint Robots*, Springer Tracts in Advanced Robotics (STAR) (Springer, Berlin, Heidelberg 2008)

9.44 C. Ott, A. Albu-Schäffer, A. Kugi, G. Hirzinger: On the passivity based impedance control of flexible joint robots, IEEE Trans. Robotics **24**, 416–429 (2008)

9.45 H. Bruyninckx, J. De Schutter: Specification of Force-controlled actions in the *task frame formalism* – A synthesis, IEEE Trans. Robotics Autom. **12**, 581–589 (1996)

9.46 H. Lipkin, J. Duffy: Hybrid twist and wrench control for a robotic manipulator, ASME J. Mech. Design **110**, 138–144 (1988)

9.47 J. Duffy: The fallacy of modern hybrid control theory that is based on *orthogonal complements* of twist and wrench spaces, J. Robotics Syst. **7**, 139–144 (1990)

9.48 K.L. Doty, C. Melchiorri, C. Bonivento: A theory of generalized inverses applied to robotics, Int. J. Robotics Res. **12**, 1–19 (1993)

9.49 T. Patterson, H. Lipkin: Duality of constrained elastic manipulation, Proc. IEEE Conf. Robotics Autom., Sacramento (1991) pp. 2820–2825

9.50 J. De Schutter, H. Bruyninckx, S. Dutré, J. De Geeter, J. Katupitiya, S. Demey, T. Lefebvre: Estimation first-order geometric parameters and monitoring contact transitions during force-controlled compliant motions, Int. J. Robotics Res. **18**(12), 1161–1184 (1999)

9.51 T. Lefebvre, H. Bruyninckx, J. De Schutter: Polyedral contact formation identification for auntonomous compliant motion, IEEE Trans. Robotics Autom. **19**, 26–41 (2007)

9.52 J. De Schutter, T. De Laet, J. Rutgeerts, W. Decré, R. Smits, E. Aerbeliën, K. Claes, H. Bruyninckx: Constraint-based task specification and estimation for sensor-based robot systems in the presence of geometric uncertainty, Int. J. Robotics Res. **26**(5), 433–455 (2007)

9.53 A. De Luca, C. Manes: Modeling robots in contact with a dynamic environment, IEEE Trans. Robotics Autom. **10**, 542–548 (1994)

9.54 T. Yoshikawa, T. Sugie, N. Tanaka: Dynamic hybrid position/force control of robot manipulators – Controller design and experiment, IEEE J. Robotics Autom. **4**, 699–705 (1988)

9.55 J. De Schutter, D. Torfs, H. Bruyninckx, S. Dutré: Invariant hybrid force/position control of a velocity controlled robot with compliant end effector using modal decoupling, Int. J. Robotics Res. **16**(3), 340–356 (1997)

9.56 R. Lozano, B. Brogliato: Adaptive hybrid force-position control for redundant manipulators, IEEE Trans. Autom. Control **37**, 1501–1505 (1992)

9.57 L.L. Whitcomb, S. Arimoto, T. Naniwa, F. Ozaki: Adaptive model-based hybrid control if geometrically constrained robots, IEEE Trans. Robotics Autom. **13**, 105–116 (1997)

9.58 B. Yao, S.P. Chan, D. Wang: Unified formulation of variable structure control schemes for robot manipulators, IEEE Trans. Autom. Control **39**, 371–376 (1992)

9.59 S. Chiaverini, B. Siciliano, L. Villani: Force/position regulation of compliant robot manipulators, IEEE Trans. Autom. Control **39**, 647–652 (1994)

9.60 J.T.-Y. Wen, S. Murphy: Stability analysis of position and force control for robot arms, IEEE Trans. Autom. Control **36**, 365–371 (1991)

9.61 R. Volpe, P. Khosla: A theoretical and experimental investigation of explicit force control strategies for manipulators, IEEE Trans. Autom. Control **38**, 1634–1650 (1993)

9.62 L.S. Wilfinger, J.T. Wen, S.H. Murphy: Integral force control with robustness enhancement, IEEE Control Syst. Mag. **14**(1), 31–40 (1994)

9.63 S. Katsura, Y. Matsumoto, K. Ohnishi: Modeling of force sensing and validation of disturbance observer for force control, IEEE Trans. Ind. Electron. **54**, 530–538 (2007)

9.64 A. Stolt, M. Linderoth, A. Robertsson, R. Johansson: Force controlled robotic assembly without a force sensor, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (2012) pp. 1538–1543

9.65 S.D. Eppinger, W.P. Seering: Introduction to dynamic models for robot force control, IEEE Control Syst. Mag. **7**(2), 48–52 (1987)

9.66 C.H. An, J.M. Hollerbach: The role of dynamic models in Cartesian force control of manipulators, Int. J. Robotics Res. **8**(4), 51–72 (1989)

9.67 R. Volpe, P. Khosla: A theoretical and experimental investigation of impact control for manipulators, Int. J. Robotics Res. **12**, 351–365 (1993)

9.68 J.K. Mills, D.M. Lokhorst: Control of robotic manipulators during general task execution: A discontinuous control approach, Int. J. Robotics Res. **12**, 146–163 (1993)

9.69 T.-J. Tarn, Y. Wu, N. Xi, A. Isidori: Force regulation and contact transition control, IEEE Control Syst. Mag. **16**(1), 32–40 (1996)

9.70 B. Brogliato, S. Niculescu, P. Orhant: On the control of finite dimensional mechanical systems with unilateral constraints, IEEE Trans. Autom. Control **42**, 200–215 (1997)

Avidbots Ex1016
Page 264 of 2154

Multimedia Contents



10. Redundant Robots

Stefano Chiaverini, Giuseppe Oriolo, Anthony A. Maciejewski

This chapter focuses on redundancy resolution schemes, i. e., the techniques for exploiting the redundant degrees of freedom in the solution of the inverse kinematics problem. This is obviously an issue of major relevance for motion planning and control purposes.

In particular, task-oriented kinematics and the basic methods for its inversion at the velocity (first-order differential) level are first recalled, with a discussion of the main techniques for handling kinematic singularities. Next, different first-order methods to solve kinematic redundancy are arranged in two main categories, namely those based on the optimization of suitable performance criteria and those relying on the augmentation of the task space. Redundancy resolution methods at the acceleration (second-order differential) level are then considered in order to take into account dynamics issues, e.g., torque minimization. Conditions under which a cyclic task motion results in a cyclic joint motion are also discussed; this is a major issue when a redundant manipulator is used to execute a repetitive task, e.g., in industrial applications. The use of kinematic redundancy for fault tolerance is analyzed in detail. Suggestions for further reading are given in a final section.

10.1 **Overview** 221
10.2 **Task-Oriented Kinematics** 224
10.2.1 Task-Space Formulation 224
10.2.2 Singularities 225
10.3 **Inverse Differential Kinematics** 227
10.3.1 The General Solution 227
10.3.2 Singularity Robustness 227
10.3.3 Joint Trajectory Reconstruction ... 231
10.4 **Redundancy Resolution via Optimization** 232
10.4.1 Performance Criteria 232
10.4.2 Local Optimization 232
10.4.3 Global Optimization 233
10.5 **Redundancy Resolution via Task Augmentation** 233
10.5.1 Extended Jacobian 233
10.5.2 Augmented Jacobian 234
10.5.3 Algorithmic Singularities 234
10.5.4 Task Priority 235
10.6 **Second-Order Redundancy Resolution** .. 236
10.7 **Cyclicity** 237
10.8 **Fault Tolerance** 237
10.9 **Conclusion and Further Reading** 239
Video-References 239
References 240

10.1 Overview

Kinematic redundancy occurs when a robotic manipulator has more degrees of freedom than those strictly required to execute a given task. This means that, in principle, no manipulator is inherently redundant; rather, there are certain tasks with respect to which it may become redundant. Because it is widely recognized that a fully general spatial task consists of following an end-effector motion trajectory requiring six degrees of freedom, a robot arm with seven or more joints is frequently considered as a typical example of an inherently redundant manipulator. However, even robot arms with fewer degrees of freedom, like conventional six-joint industrial manipulators, may become kinematically redundant for specific tasks, such as simple end-effector positioning without constraints on the orientation.

The motivation for introducing kinematic redundancy in the mechanical structure of a manipulator goes beyond that for using redundancy in traditional engineering design, namely, increasing robustness to faults so as to improve reliability (e.g., redundant processors

or sensors). In fact, endowing robotic manipulators with kinematic redundancy is mainly aimed at increasing dexterity, as is the case with a human arm.

The minimal-complexity approach that characterized early manipulator designs had the objective of minimizing cost and maintenance. For example, this led to the development of selective compliance assembly robot arm (SCARA) robots for pick-and-place operations where products had been designed for assembly, i. e., using a single axis of insertion. However, giving a manipulator the minimum number of joints required to execute its task results in a serious limitation in real-world applications where, in addition to the singularity problem, joint limits or workspace obstacles are present. All of these give rise to forbidden regions in the joint space that must be avoided during operation, thus requiring a carefully structured (and static) workspace where the motion of the manipulator can be planned in advance; this is the typical situation for workcells in traditional industrial applications.

On the other hand, the presence of additional degrees of freedom besides those strictly required to execute the task allows motions of the manipulator that do not displace the end effector (the so-called self-motions or internal motions); this implies that the same task at the end-effector level can be executed in several ways at the joint level, giving the possibility of avoiding the forbidden regions and ultimately resulting in a more versatile mechanism. Such a feature is key to allowing operation in unstructured or dynamically vary-



Fig. 10.1 The 7-DOF DLR Lightweight Robot

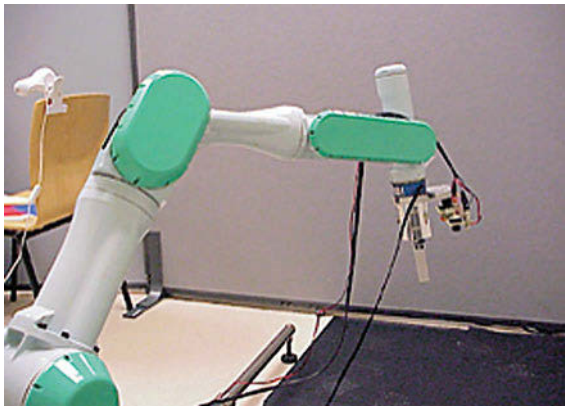


Fig. 10.2 The 7-DOF Mitsubishi PA-10 manipulator

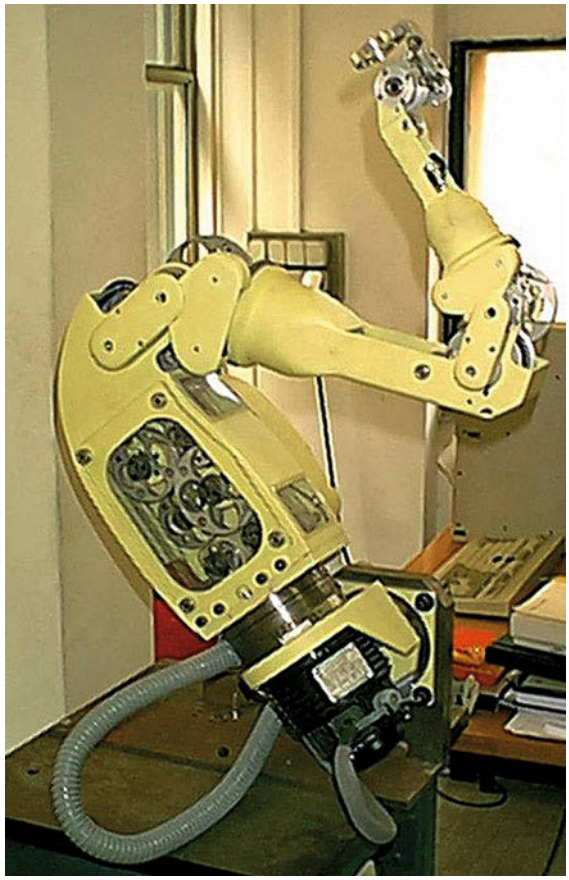


Fig. 10.3 The 8-DOF DEXTER by Scienzia Machinale

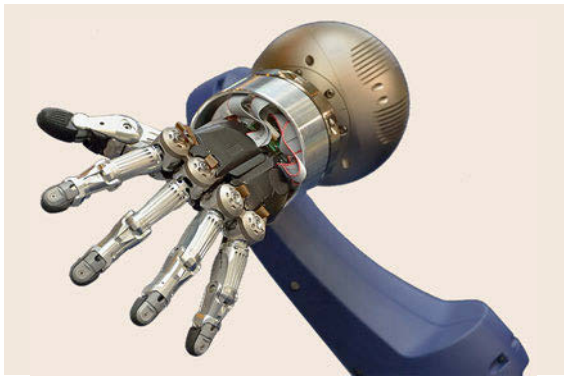


Fig. 10.4 A 5-finger hand by Schunk

ing environments that characterize advanced industrial applications and service robotics scenarios.

In practice, if properly managed, the increased dexterity characterizing kinematically redundant robots may allow them to not only avoid singularities, joint limits, and workspace obstacles, but also to minimize torque/energy over a given task, ultimately meaning that the robotic manipulator can achieve a higher degree of autonomy.

The biological archetype of a kinematically redundant manipulator is the human arm, which, not surprisingly, also inspires the terminology used to characterize the structure of serial-chain manipulators. In fact, the human arm has three degrees of freedom at the *shoulder*, one degree of freedom at the *elbow* and three degrees of freedom at the *wrist*. The available redundancy can be easily verified by locking one's wrist, e.g., on a table and moving the elbow while keeping the shoulder still. The kinematic arrangement of the human arm has been replicated in a number of robots often termed as *human-arm-like* manipulators. This family of 7-DOF (degree of freedom) manipulators includes the DLR (Deutsches Zentrum für Luft- und Raumfahrt) Lightweight Robot (Fig. 10.1) and the Mitsubishi PA-10 robot (Fig. 10.2). An example of an 8-DOF robot is the DEXTER by Scienzia Machinale (Fig. 10.3). Manipulators with a larger number of joints are often called *hyperredundant* robots, and include many snake-like robots described in the literature.

The use of two or more robotic structures to execute a task (as in the case of multifingered hands, humanoid mockups or cooperating manipulators, Figs. 10.4–10.7, respectively) also gives rise to kinematic redundancy. Redundant mechanisms also include vehicle-manipulator systems (Fig. 10.8); in this case, however, the possible presence of nonholonomic constraints on the



Fig. 10.5 The Baxter working robot by Rethink Robotics



Fig. 10.6 The NASA (National Aeronautics and Space Administration) Robonaut



Fig. 10.7 A cooperating dual-arm concept robot by Asea Brown Boveri (ABB)



Fig. 10.8 A huge vehicle-manipulator system by KUKA Aktiengesellschaft

motion of the base must be properly taken into account in order to determine the actual degree of redundancy.

Although the realization of a kinematically redundant structure raises a number of issues from the point of view of mechanical design, this chapter focuses on redundancy resolution schemes, i. e., techniques for exploiting the redundant degrees of freedom in the solution of the inverse kinematics problem. This is an issue of major relevance for motion planning and control purposes.

10.2 Task-Oriented Kinematics

The relationship between the variables representing the configuration of an articulated manipulator and those describing an assigned task in the appropriate space can be established at the position, velocity or acceleration level. In particular, consideration of the first-order task kinematics inevitably results in discussion of the task Jacobian matrix, which is central to many redundancy resolution techniques.

10.2.1 Task-Space Formulation

A manipulator consists of a chain of rigid bodies articulated by joints. If q_i denotes the variable characterizing the relative displacement of body i with respect to body $i-1$, the vector $\boldsymbol{q} = (q_1 \dots q_N)^{\mathrm{T}}$ uniquely describes the configuration of an N-joint serial-chain manipulator. Joint i may be either prismatic or revolute, in which case q_i measures the relative translation or rotation of the attached links, respectively.

While the manipulator is naturally described and actuated in the *joint space*, its operation is conveniently specified in terms of the vector $\boldsymbol{t} = (t_1 \dots t_M)^{\mathrm{T}}$, which typically defines the location of the manipulator end-effector in a suitably defined *task space*. In the general case, it is of dimension $M = 6$ and $\boldsymbol{t}$ is chosen so that its first three components represent the position of the end effector, while its last three components represent a minimal description of the end-effector orientation (such as Euler angles or the roll–pitch–yaw representation), i. e.,

$$\boldsymbol{t} = \begin{bmatrix} p_x & p_y & p_z & \alpha & \beta & \gamma \end{bmatrix}^{\mathrm{T}} .$$

Typically, one has $N \geq M$, so that the joints can provide at least the number of degrees of freedom required for the end-effector task. If $N > M$ strictly, the manipulator is *kinematically redundant*.

The relationship between the joint-space coordinate vector $\boldsymbol{q}$ and the task-space coordinate vector $\boldsymbol{t}$ is expressed by the *direct kinematics* equation

$$\boldsymbol{t} = \boldsymbol{k}_{\mathrm{t}}(\boldsymbol{q}) , \tag{10.1}$$

where $\boldsymbol{k}_{\mathrm{t}}$ is a nonlinear vector function.

Task Jacobian and Geometric Jacobian

It is useful to consider the *first-order differential kinematics* [10.1]

$$\dot{\boldsymbol{t}} = \mathbf{J}_{\mathrm{t}}(\boldsymbol{q})\, \dot{\boldsymbol{q}} , \tag{10.2}$$

that can be obtained by differentiating (10.1) with respect to time. In (10.2), $\dot{\boldsymbol{t}}$ is the task-space velocity vector, $\dot{\boldsymbol{q}}$ is the joint-space velocity vector, and $\mathbf{J}_{\mathrm{t}}(\boldsymbol{q}) = \partial \boldsymbol{k}_{\mathrm{t}} / \partial \boldsymbol{q}$ is the $M \times N$ *task Jacobian* matrix (also called the *analytic* Jacobian).

Remarkably, the components of $\dot{\boldsymbol{t}}$ relative to the end-effector orientation express the rate of change of the parameters characterizing the adopted minimal representation; they are not the components of the angular velocity vector of the end-effector. Indeed, denoting by $\mathbf{v}_N$ the 3×1 translational velocity vector and by $\boldsymbol{\omega}_N$ the 3×1 angular velocity vector of the end-effector, and defining the end-effector spatial velocity $\boldsymbol{v}_N$ as

$$\boldsymbol{v}_N = \begin{pmatrix} \mathbf{v}_N \\ \boldsymbol{\omega}_N \end{pmatrix} , \tag{10.3}$$

the following relationship holds

$$\dot{\boldsymbol{t}} = \mathbf{T}(\boldsymbol{t})\, \boldsymbol{v}_N , \tag{10.4}$$

where $\mathbf{T}$ is an $M \times 6$ transformation matrix that is a function of $\boldsymbol{t}$ only. In the case of $M = 6$, the transformation matrix $\mathbf{T}$ takes the form

$$\mathbf{T} = \begin{pmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{R} \end{pmatrix} , \tag{10.5}$$

where $\mathbf{I}$ and $\mathbf{0}$ are, respectively, the identity and null matrix of proper dimensions, and $\mathbf{R}$ is a 3×3 matrix that specifically depends on the minimal representation used to describe the end-effector orientation.

For a given manipulator, the mapping

$$\boldsymbol{v}_N = \mathbf{J}(\boldsymbol{q})\, \dot{\boldsymbol{q}} \tag{10.6}$$

relates a joint-space velocity to the corresponding end-effector velocity through the $6 \times N$ *geometric Jacobian*

matrix $\mathbf{J}$. The geometric Jacobian matrix is of major concern in the kinematic analysis of a manipulator because it allows description of the motion capabilities of the end-effector (in terms of its free rigid-body spatial velocity) as a result of the velocity commands at the joints in the current configuration. As a matter of fact, if $\boldsymbol{\mathcal{J}}_{Pi}\dot{q}_i$ denotes the contribution of $\dot{q}_i$ to $\mathbf{v}_N$, z_{i-1} is the unit vector of the axis of joint i and $\boldsymbol{r}_{i-1,e}$ is the vector expressing the position of the end effector frame with respect to the joint i frame, it is

$$\boldsymbol{\mathcal{J}}_{Pi} = \begin{cases} z_{i-1} & \text{if joint } i \text{ is prismatic} \\ z_{i-1} \times \boldsymbol{r}_{i-1,e} & \text{if joint } i \text{ is revolute} \end{cases},$$

whereas if $\boldsymbol{\mathcal{J}}_{Oi}\dot{q}_i$ denotes the contribution of $\dot{q}_i$ to $\boldsymbol{\omega}_N$, it is

$$\boldsymbol{\mathcal{J}}_{Oi} = \begin{cases} \mathbf{0} & \text{if joint } i \text{ is prismatic} \\ z_{i-1} & \text{if joint } i \text{ is revolute} \end{cases}.$$

By comparing (10.2), (10.4), and (10.6), the relation between the geometric Jacobian and the task Jacobian can be found as

$$\mathbf{J}_{\mathrm{t}}(\boldsymbol{q}) = \mathbf{T}(\boldsymbol{t})\,\mathbf{J}(\boldsymbol{q})\,. \tag{10.7}$$

Second-Order Differential Kinematics

While the first-order differential kinematics (10.2) relates task-space to joint-space velocities, further differentiation with respect to time provides an analogous relationship between accelerations

$$\ddot{\boldsymbol{t}} = \mathbf{J}_{\mathrm{t}}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \dot{\mathbf{J}}_{\mathrm{t}}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}}\,. \tag{10.8}$$

This equation is also known as the *second-order differential kinematics*.

10.2.2 Singularities

In this section the occurrence of singular configurations is considered to properly analyze their effect on inverse kinematics solutions.

Representation and Kinematic Singularities

A robot configuration $\boldsymbol{q}$ is *singular* if the task Jacobian matrix $\mathbf{J}_{\mathrm{t}}$ is rank-deficient there. Considering the role of $\mathbf{J}_{\mathrm{t}}$ in (10.2) and (10.8), it is easy to realize that at a singular configuration it is impossible to generate end-effector task velocities or accelerations in certain directions. Further insight can be gained by looking at (10.7), which indicates that a singularity may be due to a loss of rank of the transformation matrix $\mathbf{T}$ and/or of the geometric Jacobian matrix $\mathbf{J}$.

Rank deficiencies of $\mathbf{T}$ are only related to the mathematical relationship established by $\mathbf{R}$ between the angular velocity vector of the end-effector and the components of $\dot{\boldsymbol{t}}$ relative to the end-effector orientation; because the expression of $\mathbf{R}$ depends on the adopted minimal representation of the orientation, a configuration at which $\mathbf{T}$ is singular is then referred to as a *representation singularity*. Remarkably, any minimal description of the end-effector orientation experiences the occurrence of representation singularities. Furthermore, a given configuration may or may not yield a representation singularity depending on which description of orientation is used.

A representation singularity is not directly related to the true motion capabilities of the manipulator structure, which can instead be inferred by the analysis of the geometric Jacobian matrix. Rank deficiencies of $\mathbf{J}$ are in fact related to loss of mobility of the manipulator end-effector; indeed, end-effector velocities exist in this case that are unfeasible for any velocity commanded at the joints. A configuration at which $\mathbf{J}$ is singular is referred to as a *kinematic singularity*.

Because this chapter focuses on the inversion of the task differential kinematics (10.2) and (10.8), in the following the task Jacobian matrix and its singularities (which include representation and kinematic singularities) are studied in detail. The case $N \geq M$ is considered, which encompasses conventional as well as redundant manipulators.

Singular Value Decomposition of the Jacobian

To analyze the linear mapping (10.2), the *singular value decomposition* (SVD) of the Jacobian matrix is adopted; remarkably, this powerful numerical tool is the sole reliable method to compute the rank of a matrix and to study near-singular linear mappings. The classic *Golub–Reinsch* algorithm [10.2], which is the most efficient and numerically stable algorithm to compute the SVD of an arbitrary matrix, may however be computationally demanding in view of real-time applications. A faster algorithm that takes advantage of the nature of robotic matrix calculations has been proposed [10.3]; this makes it possible to improve real-time kinematic control techniques.

The SVD of the task Jacobian matrix can be written in the form

$$\mathbf{J}_{\mathrm{t}} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^{\mathrm{T}} = \sum_{i=1}^{M} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^{\mathrm{T}}\,, \tag{10.9}$$

where $\mathbf{U}$ is the $M \times M$ orthonormal matrix of the output singular vectors $\boldsymbol{u}_i$, $\mathbf{V}$ is the $N \times N$ orthonormal matrix of the input singular vectors $\boldsymbol{v}_i$, and $\boldsymbol{\Sigma} = (\mathbf{S}\ \mathbf{0})$ is the $M \times N$ matrix whose $M \times M$ diagonal submatrix $\mathbf{S}$

Avidbots Ex1016
Page 269 of 2154

contains the singular values σ_i of the matrix $\mathbf{J}_\mathrm{t}$. It is worth noting that the SVD is a well-behaved function of its matrix argument; therefore, the singular values and associated subspaces do not change radically in the neighborhood of the current configuration. Letting rank $(\mathbf{J}_\mathrm{t}) = R$, the following hold

$$\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_R > \sigma_{R+1} = \cdots = 0 \,,$$
$$\mathcal{R}(\mathbf{J}_\mathrm{t}) = \mathrm{span}\{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_R\} \,,$$
$$\mathcal{N}(\mathbf{J}_\mathrm{t}) = \mathrm{span}\{\boldsymbol{v}_{R+1}, \ldots, \boldsymbol{v}_N\} \,.$$

If the task Jacobian is full-rank ($R = M$), all the singular values are nonzero, the range space of $\mathbf{J}_\mathrm{t}$ is the entire $\mathbb{R}^M$, and the null space of $\mathbf{J}_\mathrm{t}$ has dimension $N-M$. In a singular configuration, $R < M$; thus, the last $M-R$ singular values are zero, the range space of $\mathbf{J}_\mathrm{t}$ is an R-dimensional subspace of $\mathbb{R}^M$, and the dimension of the null space of $\mathbf{J}_\mathrm{t}$ increases to $N-R$. An interpretation of this from a kinematic viewpoint is presented in the following.

Feasible Velocities. At each configuration, the range space of $\mathbf{J}_\mathrm{t}$ is the set of task-space velocities that can be obtained as a result of all possible joint-space velocities $\dot{\boldsymbol{q}}$; therefore, it constitutes the so-called subspace of *feasible velocities* for the end-effector task. A basis for $\mathcal{R}(\mathbf{J}_\mathrm{t})$ is given by the first R output singular vectors; accordingly, the effect of a singularity is to decrease the dimension of the range space of $\mathbf{J}_\mathrm{t}$ that corresponds to a reduction of the space of feasible velocities.

Null-Space Velocities. At each configuration, the null space of $\mathbf{J}_\mathrm{t}$ is the set of joint-space velocities that yield zero task velocity; these are thus shortly called *null-space velocities*. A basis for $\mathcal{N}(\mathbf{J}_\mathrm{t})$ is given by the $N-R$ last input singular vectors, which represent independent linear combinations of the velocities at each joint. From this viewpoint, the effect of a singularity is to increase the dimension of the null space of $\mathbf{J}_\mathrm{t}$ by introducing a further independent linear combination of joint velocities that produces a zero task velocity.

According to (10.2) and (10.9), a joint velocity along the i-th input singular vector results in a task velocity that lies along the i-th output singular vector

$$\forall \rho \in \mathbb{R} \quad \dot{\boldsymbol{q}} = \rho\, \boldsymbol{v}_i \quad \Rightarrow \quad \boldsymbol{t} = \sigma_i \rho\, \boldsymbol{u}_i \,.$$

Thus, the i-th singular value of $\mathbf{J}_\mathrm{t}$ can be viewed as a gain factor relating motion along the $\boldsymbol{v}_i$ direction of the joint velocity space to the resulting motion along the $\boldsymbol{u}_i$ direction of the task velocity space. When a singularity is approached, the R-th singular value σ_R tends to zero and the task velocity produced by a fixed joint velocity along $\boldsymbol{v}_R$ is decreased proportionally. At a singular configuration, the joint-space velocity along $\boldsymbol{v}_R$ belongs to the null-space velocities and task velocities along $\boldsymbol{u}_R$ become unfeasible for the manipulator.

In the generic case, the joint-space velocity $\dot{\boldsymbol{q}}$ is an arbitrary linear combination of individual joint velocities with nonzero components along all the $\boldsymbol{v}_i$. Its effect can be analyzed by combining the single effects of the above components. Remarkably, the components of $\dot{\boldsymbol{q}}$ in the null space of $\mathbf{J}_\mathrm{t}$ produce a change in the configuration of the manipulator without affecting its task velocity. This can be exploited to achieve additional goals – such as obstacle or singularity avoidance – in addition to the realization of a desired task motion, and constitutes the core of redundancy resolution approaches.

Distance from Singularities

Clearly, the effect of a singularity is experienced not only at the singular configuration itself but also in its neighborhood (Sect. 10.3.2). For this reason, it is important to be able to characterize the distance from singularities through suitable measures; these can then be exploited to counteract undesirable effects.

Because each singularity is associated with a reduction in the rank of $\mathbf{J}_\mathrm{t}$, one conceptually simple possibility in the case of a square Jacobian matrix ($M = N$) is to compute its determinant. A generalization of this idea that works also for nonsquare Jacobians is the *manipulability measure* [10.4], defined as

$$\mu = \sqrt{|\mathbf{J}_\mathrm{t}\, \mathbf{J}_\mathrm{t}^\mathrm{T}|} \,.$$

It can be recognized that the manipulability measure is equal to the product of the singular values of $\mathbf{J}_\mathrm{t}$, i. e.,

$$\mu = \prod_{i=1}^{M} \sigma_i \,,$$

and thus its zeros coincide with the singularities.

Another possible measure of distance from a singular configuration is the *condition number* of the Jacobian matrix [10.5], defined as

$$\kappa = \frac{\sigma_1}{\sigma_M} \,.$$

The condition number has values ranging from 1, at configurations where all the singular values are equal, to ∞, at singular configurations. Note that when $\kappa = 1$ the end effector has the same motion capability in all task space directions because all the singular values are equal – i. e., the arm is at an *isotropic configuration* – whereas at a singularity it loses mobility in some task-space direction.

An even more direct measure of the distance from singular configurations is the *smallest singular value* of the Jacobian matrix [10.5], i. e.,

$$\sigma_{\min} = \sigma_M \, .$$

A computationally light estimate of the smallest singular value can be obtained either via numerical methods [10.3, 6, 7] or based on a kinematic analysis of the robot structure [10.8].

It must be noted that the manipulability measure may remain constant even in the presence of variations of either the condition number or the smallest singular value of $\mathbf{J}_t$. On the other hand, because the smallest singular value changes more radically near singularities than the other singular values, it dominates the behavior of the determinant and the condition number of the Jacobian matrix; therefore, the most effective measure of distance from singular configurations is the smallest singular value of $\mathbf{J}_t$ [10.5].

10.3 Inverse Differential Kinematics

In order to accomplish a task, a proper joint motion must be commanded to the manipulator; therefore, it is necessary to derive mathematical relations that allow one to compute joint-space variables corresponding to the assigned task-space variables. This is the objective of the inverse kinematics problem.

The inverse kinematics problem can be solved by inverting the direct kinematics equation (10.1), the first-order differential kinematics (10.2) or the second-order differential kinematics (10.8). If the task is time-varying (i. e., if a desired trajectory $\boldsymbol{t}(t)$ is assigned), it is convenient to solve the differential kinematic relationships because these represent linear equations using the task Jacobian matrix [10.9].

10.3.1 The General Solution

Under the assumption that the manipulator is kinematically redundant (i. e., $M < N$), one solution of (10.2) or (10.8) can be expressed by resorting to the pseudoinverse $\mathbf{J}_t^\dagger$ of the task Jacobian matrix [10.1, 10]; this is the unique matrix satisfying the Moore–Penrose conditions [10.11–13]

$$\begin{aligned}
&\mathbf{J}_t \, \mathbf{J}_t^\dagger \mathbf{J}_t = \mathbf{J}_t \, , \\
&\mathbf{J}_t^\dagger \mathbf{J}_t \, \mathbf{J}_t^\dagger = \mathbf{J}_t^\dagger \, , \\
&(\mathbf{J}_t \, \mathbf{J}_t^\dagger)^{\mathrm{T}} = \mathbf{J}_t \, \mathbf{J}_t^\dagger \, , \\
&(\mathbf{J}_t^\dagger \mathbf{J}_t)^{\mathrm{T}} = \mathbf{J}_t^\dagger \mathbf{J}_t \, .
\end{aligned} \tag{10.10}$$

If $\mathbf{J}_t$ is low-rectangular and full-rank, its pseudoinverse can be expressed as

$$\mathbf{J}_t^\dagger = \mathbf{J}_t^{\mathrm{T}} (\mathbf{J}_t \, \mathbf{J}_t^{\mathrm{T}})^{-1} \, . \tag{10.11}$$

If $\mathbf{J}_t$ is square, expression (10.11) reduces to the standard inverse matrix.

The general solution of (10.2) can be written as

$$\dot{\boldsymbol{q}} = \mathbf{J}_t^\dagger \dot{\boldsymbol{t}} + (\mathbf{I} - \mathbf{J}_t^\dagger \mathbf{J}_t) \, \dot{\boldsymbol{q}}_0 \, . \tag{10.12}$$

Here, $\mathbf{I} - \mathbf{J}_t^\dagger \mathbf{J}_t$ represents the orthogonal projection matrix into the null space of $\mathbf{J}_t$, and $\dot{\boldsymbol{q}}_0$ is an arbitrary joint-space velocity; the second part of the solution is therefore a null-space velocity. Equation (10.12) provides all least-squares solution to the end-effector task constraint (10.2), i. e., it minimizes $\|\dot{\boldsymbol{t}} - \mathbf{J}\dot{\boldsymbol{q}}\|$. In particular, if $\mathbf{J}_t$ is low-rectangular and full-rank, all joint velocities in the form (10.12) exactly realize the assigned task velocity. By acting on $\dot{\boldsymbol{q}}_0$, one can still obtain different joint velocities that give the same end-effector task velocity; therefore, as will be discussed later in detail, solutions in the form of (10.12) are typically used to represent redundancy resolution techniques.

The particular solution obtained by setting $\dot{\boldsymbol{q}}_0 = 0$ in (10.12),

$$\dot{\boldsymbol{q}} = \mathbf{J}_t^\dagger \, \dot{\boldsymbol{t}} \, , \tag{10.13}$$

provides the least-squares solution of (10.2) with minimum norm, and is known as the *pseudoinverse solution*. In terms of the inverse differential kinematics problem, the least-squares property quantifies the accuracy of the end-effector task realization, while the minimum norm property may be relevant for the feasibility of the joint-space velocities.

As for the second-order kinematics (10.8), its least-squares solutions can be expressed in the general form

$$\ddot{\boldsymbol{q}} = \mathbf{J}_t^\dagger (\ddot{\boldsymbol{t}} - \dot{\mathbf{J}}_t \dot{\boldsymbol{q}}) + (\mathbf{I} - \mathbf{J}_t^\dagger \mathbf{J}_t) \, \ddot{\boldsymbol{q}}_0 \, , \tag{10.14}$$

where $\ddot{\boldsymbol{q}}_0$ is an arbitrary joint-space acceleration. As above, choosing $\ddot{\boldsymbol{q}}_0 = 0$ in (10.14) gives the minimum-norm acceleration solution

$$\ddot{\boldsymbol{q}} = \mathbf{J}_t^\dagger (\ddot{\boldsymbol{t}} - \dot{\mathbf{J}}_t \, \dot{\boldsymbol{q}}) \, . \tag{10.15}$$

10.3.2 Singularity Robustness

We now investigate the kinematics aspects involved in the first-order inverse mappings (10.12) and (10.13)

with respect to the handling of singularities. With reference to the SVD of $\mathbf{J}_t$ in (10.9), let us consider the following decomposition of the matrix $\mathbf{J}_t^\dagger$

$$\mathbf{J}_t^\dagger = \mathbf{V}\boldsymbol{\Sigma}^\dagger\mathbf{U}^T = \sum_{i=1}^{R} \frac{1}{\sigma_i}\boldsymbol{v}_i\boldsymbol{u}_i^T \,, \tag{10.16}$$

where, as above, R denotes the rank of the task Jacobian matrix. Analogously to (10.9), the following hold

$$\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_R > \sigma_{R+1} = \cdots = 0 \,,$$
$$\mathcal{R}(\mathbf{J}_t^\dagger) = \mathcal{N}^\perp(\mathbf{J}_t) = \text{span}\{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_R\} \,,$$
$$\mathcal{N}(\mathbf{J}_t^\dagger) = \mathcal{R}^\perp(\mathbf{J}_t) = \text{span}\{\boldsymbol{u}_{R+1}, \ldots, \boldsymbol{u}_M\} \,.$$

Notice that, if the Jacobian matrix is full-rank, the range space of $\mathbf{J}_t^\dagger$ is an M-dimensional subspace of $\mathbb{R}^N$ and the null space of $\mathbf{J}_t^\dagger$ is trivial. In a singular configuration ($R < M$), the range space of $\mathbf{J}_t^\dagger$ is an R-dimensional subspace of $\mathbb{R}^N$, and an $M-R$-dimensional null space of $\mathbf{J}_t^\dagger$ exists.

The range space of $\mathbf{J}_t^\dagger$ is the set of joint-space velocities $\dot{\boldsymbol{q}}$ that can be obtained via the inverse kinematic mapping (10.13) as a result of all possible task velocities $\dot{\boldsymbol{t}}$. Because these $\dot{\boldsymbol{q}}$ belong to the orthogonal complement of the null space of $\mathbf{J}_t$, the pseudoinverse solution (10.13) satisfies the least-squares condition, as expected.

The null space of $\mathbf{J}_t^\dagger$ is the set of the task velocities $\dot{\boldsymbol{t}}$ that yield null joint-space velocity in the current configuration; these $\dot{\boldsymbol{t}}$ belong to the orthogonal complement of the space of feasible task velocities. Therefore, one effect of the pseudoinverse solution (10.13) is to filter out the unfeasible components of the commanded task velocity while allowing exact tracking of the feasible components; this is related to the minimum-norm property.

If the assigned task velocity is aligned with $\boldsymbol{u}_i$, the corresponding joint-space velocity – computed via (10.13) – is obtained along $\boldsymbol{v}_i$ modulated by the factor $1/\sigma_i$. When a singularity is approached, the R-th singular value tends to zero and a fixed task velocity command along $\boldsymbol{u}_R$ requires joint-space velocities along $\boldsymbol{v}_R$ that grow unboundedly in proportion to the factor $1/\sigma_R$. At the singular configuration, the $\boldsymbol{u}_R$ direction becomes unfeasible for the task variables and $\boldsymbol{v}_R$ adds to the null-space velocities of the arm.

According to the above, two main problems are inherently related to the basic inverse differential kinematics solution (10.13), namely:

- At near-singular configurations, excessive joint-space velocities may result, due to the component of $\dot{\boldsymbol{t}}$ along the direction which becomes unfeasible at the singularity.
- At the singular configuration, discontinuity of the joint-space solution is experienced if $\dot{\boldsymbol{t}}$ has a non-null unfeasible component.

The same is obviously true for the complete inverse solution (10.12).

Both the above problems are of major concern for kinematic control of manipulators, where the computed joint-space velocities must actually be executed by the robot arm. This has motivated the development of modified inverse differential mappings, so as to ensure proper behavior of the manipulator throughout its workspace independent of the configuration of the robot. A reasonable approach is to preserve the mapping (10.13) far from singularities and to set up local modifications of it only inside a region enclosing the singular configuration; the definition of the region depends on a suitable measure of distance from the singularity, while the modified mapping must ensure feasible and continuous joint velocities.

Modification of the Planned Trajectory

One way to tackle the problem of singularities is to act at the planning level by either designing trajectories that do not incur in unavoidable singularities or assigning task-space motion commands that are feasible for the robot arm. However, these solutions rely on perfect task planning and cannot be used in real-time sensory control applications, in which motion commands are generated online.

Because singularities are a function of configuration and not workspace location, in general, it is not possible to avoid singularities by restricting the workspace trajectory of the manipulator. There do exist cases where avoidance of singular configurations at the motion planning level is relatively simple, e.g., when dealing with those associated with workspace boundary such as the elbow singularity of an anthropomorphic arm. However, this approach may be difficult for those singularities, like the wrist one, that may occur everywhere in the workspace.

Another possibility is to perform a joint-space interpolation when the planned trajectory is close to a singularity [10.14]; however, this may cause large errors in tracking the originally assigned task-space motion.

Acting in the task space, a method based on time-scale transformation is presented in [10.15], which slows down the manipulator's motion when a singularity is approached; this technique, however, fails at the singularity.

Because a task-space robot control system must be able to guide the motion of the manipulator safely through singularities, considerable research effort has been devoted to the derivation of well-defined and

Avidbots Ex1016
Page 272 of 2154

continuous inverse kinematic mappings instead of the above ad hoc approaches.

Removal of Dependent Rows or Columns of the Jacobian Matrix

The first requirement of solution (10.13) is the availability of a general algorithm to compute the pseudoinverse of $\mathbf{J}_t$ also when the Jacobian is singular. Several techniques presented in the literature can be arranged in this framework; they consist either in removing the unfeasible end-effector reference components [10.16] or in using nonsingular blocks of the Jacobian matrix [10.10]. The main problem with this type of solution is the specification of the directions of unfeasible velocities in a systematic way and the need to have smooth transitions between the usual inverse kinematic algorithm and the algorithms used close to the singularities.

A systematic procedure to compute the pseudoinverse of the Jacobian matrix can be devised by taking advantage of a kinematic analysis of the manipulator structure because for many manipulators it is possible to identify and describe classes of singular configurations in suitably defined link-fixed frames. This approach has been demonstrated for a six-degree-of-freedom elbow geometry in [10.17, 18].

As for the continuity of the solution across a singularity, it must be observed that, while the singular vectors change very little in the neighborhood of the singularity, the term

$$\frac{1}{\sigma_M}\, \boldsymbol{v}_M \boldsymbol{u}_M^{\mathrm{T}}\, \dot{\boldsymbol{t}}$$

suddenly disappears from (10.16) when R becomes less than M at the singular configuration.

One possibility to avoid this problem is to make

$$\boldsymbol{u}_M^{\mathrm{T}}\, \dot{\boldsymbol{t}} \approx 0$$

in the neighborhood of the singularity, which means avoiding commanding task velocities along the direction that becomes unfeasible at the singular configuration. This would require some method for efficiently computing the singular subspace associated with near-zero singular value(s) as described in the upcoming section on numerical filtering.

Independently from the assigned $\dot{\boldsymbol{t}}$, continuity of the pseudoinverse solution can be ensured by treating the manipulator as singular in a suitably defined region around each singularity through a modified Jacobian $\bar{\mathbf{J}}_t$, so that $M-R$ extra degrees of freedom are available [10.17–19]; these can be used without significantly affecting the end-effector velocity, because the modified Jacobian $\bar{\mathbf{J}}_t$ approximates $\mathbf{J}_t$ inside the region.

The described approach is difficult to generalize for multiple singularities. For the typical anthropomorphic geometry of industrial manipulators, however, only the wrist singularity is of primary concern because it may occur everywhere in the workspace, while the elbow and shoulder singularities are naturally characterized in the task space and thus can be avoided during planning.

Regularization/Damped Least-Squares Technique

The use of the damped least-squares technique in the inverse differential kinematics problem has been independently proposed in [10.9, 20]. The method corresponds to solving the optimization problem

$$\min_{\dot{\boldsymbol{q}}} \left(\|\dot{\boldsymbol{t}} - \mathbf{J}_t \dot{\boldsymbol{q}}\|^2 + \lambda^2 \|\dot{\boldsymbol{q}}\|^2 \right) , \tag{10.17}$$

which results in a solution $\dot{\boldsymbol{q}}$ that minimizes the end-effector tracking error from the set of all joint velocities that do not exceed $\|\dot{\boldsymbol{q}}\|$. Here, $\lambda \in \mathbb{R}$ is the damping factor that represents the trade-off between the least-squares and the mimimum-norm properties. When λ is zero, the solution to (10.17) and (10.13) become identical.

The solution to (10.17) can be written in two equivalent forms

$$\dot{\boldsymbol{q}} = \mathbf{J}_t^{\mathrm{T}} (\mathbf{J}_t \mathbf{J}_t^{\mathrm{T}} + \lambda^2 \mathbf{I})^{-1} \dot{\boldsymbol{t}} , \tag{10.18}$$

$$\dot{\boldsymbol{q}} = (\mathbf{J}_t^{\mathrm{T}} \mathbf{J}_t + \lambda^2 \mathbf{I})^{-1} \mathbf{J}_t^{\mathrm{T}} \dot{\boldsymbol{t}} . \tag{10.19}$$

The computational load of (10.18) is lower than that of (10.19), because usually $N \geq M$. In the remainder, we will refer to the damped least-squares solution as

$$\dot{\boldsymbol{q}} = \mathbf{J}_t^{*}(\boldsymbol{q})\, \dot{\boldsymbol{t}} , \tag{10.20}$$

whenever explicit specification of the computation used is not essential.

Condition (10.17) implies consideration of accuracy and feasibility at the same time when choosing the joint-space velocity required to match the given $\dot{\boldsymbol{t}}$. In this regard it is essential to suitably select the value to be assigned to the damping factor: small values of λ give accurate solutions but low robustness to the occurrence of singular and near-singular configurations; high values of λ result in low tracking accuracy even when a feasible and accurate solution would be possible.

In the framework of singular value decomposition, the solution (10.20) can be written as

$$\dot{\boldsymbol{q}} = \sum_{i=1}^{R} \frac{\sigma_i}{\sigma_i^2 + \lambda^2}\, \boldsymbol{v}_i \boldsymbol{u}_i^{\mathrm{T}}\, \dot{\boldsymbol{t}} . \tag{10.21}$$

Remarkably, we have

$$\begin{aligned}\mathcal{R}(\mathbf{J}_t^*) &= \mathcal{R}(\mathbf{J}_t^\dagger) = \mathcal{N}^\perp(\mathbf{J}_t)\\ &= \text{span}\{\boldsymbol{v}_1,\ldots,\boldsymbol{v}_R\}\,,\\ \mathcal{N}(\mathbf{J}_t^*) &= \mathcal{N}(\mathbf{J}_t^\dagger) = \mathcal{R}^\perp(\mathbf{J}_t)\\ &= \text{span}\{\boldsymbol{u}_{R+1},\ldots,\boldsymbol{u}_M\}\,.\end{aligned}$$

As with $\mathbf{J}_t^\dagger$, if the Jacobian matrix is full-rank, the range space of $\mathbf{J}_t^*$ is an M-dimensional subspace of $\mathbb{R}^N$ and the null space of $\mathbf{J}_t^*$ is trivial; in a singular configuration, the range space of $\mathbf{J}_t^*$ is an R-dimensional subspace of $\mathbb{R}^N$, and an $M-R$-dimensional null space of $\mathbf{J}_t^*$ exists.

It is clear that, with respect to the pure least-squares solution (10.13), in (10.21) the components for which $\sigma_i \gg \lambda$ are little influenced by the damping factor, because in this case

$$\frac{\sigma_i}{\sigma_i^2+\lambda^2} \approx \frac{1}{\sigma_i}\,.$$

On the other hand, when a singularity is approached, the smallest singular value tends to zero while the associated component of the solution is driven to zero by the factor σ_i/λ^2; this progressively reduces the joint velocity required to achieve near-degenerate components of the commanded $\dot{\boldsymbol{t}}$. At the singularity, the solutions (10.20) and (10.13) behave identically as long as the remaining singular values are significantly larger than the damping factor. One can show that the maximum gain factor of $1/(2\lambda)$ relating the task velocity component along $\boldsymbol{u}_i$ to the resulting joint velocity along $\boldsymbol{v}_i$ occurs when $\sigma_i = \lambda$.

Selection of the Damping Factor. According to above, λ determines the degree of approximation introduced with respect to the pure least-squares solution given by the pseudoinverse. An optimal choice for λ requires consideration of the smallest non-null singular value experienced along the given trajectory and of the minimum damping needed to ensure feasible joint velocities.

To achieve good performance in the entire manipulator's workspace the use of a configuration-varying damping factor has been proposed [10.9]. The natural choice is to adjust λ as a function of some measure of distance from the singularity at the current configuration of the robot arm. Far from singular configurations feasible joint velocities are obtained; thus the accuracy requirement prevails and low damping must be used. Close to a singularity, task velocity commands along the unfeasible directions give large joint velocities and the accuracy requirement must be relaxed; in these cases, high damping is needed.

A first proposal was to adjust the damping factor as a function of the manipulability measure [10.9]. Because a more effective measure of distance from singular configurations is the smallest singular value of the Jacobian matrix [10.5], its use has been considered later for building a variable damping factor.

If an estimate $\tilde{\sigma}_M$ of the smallest singular value is available, the following choice for the damping factor can be adopted [10.21]

$$\lambda^2 = \begin{cases} 0 & \text{when } \tilde{\sigma}_M \geq \varepsilon \\ \left[1-\left(\frac{\tilde{\sigma}_M}{\varepsilon}\right)^2\right]\lambda_{\max}^2 & \text{otherwise,}\end{cases} \qquad (10.22)$$

which ensures continuity and good shaping of the solution. In (10.22), ε defines the size of the singular region, in which damping is used, and $\lambda_{\max}$ sets the maximum value of the damping factor, which occurs just at the singularity.

Numerical Filtering. As can be recognized from (10.21), the damping factor affects the accuracy of the solution along each end-effector velocity component; however, the sole unfeasible direction is responsible for the (possible) loss of tracking ability. To overcome this problem, selective filtering of the end-effector velocity components was proposed in [10.6]. The method can be generalized as follows; if estimates $\tilde{\boldsymbol{u}}_i$ of the output singular vectors associated with the $M-K$ smallest singular values – which give the components to become unfeasible – are available, the solution can be written in the form

$$\dot{\boldsymbol{q}} = \mathbf{J}_t^{\mathrm{T}}\left(\mathbf{J}_t\,\mathbf{J}_t^{\mathrm{T}} + \lambda^2\mathbf{I} + \beta^2\sum_{i=K+1}^{M}\tilde{\boldsymbol{u}}_i\,\tilde{\boldsymbol{u}}_i^{\mathrm{T}}\right)^{-1}\dot{\boldsymbol{t}}\,, \qquad (10.23)$$

where β provides the largest contribution to damping only along the unfeasible components. This solution is an approximation to applying a greater selective filtering to the near singular components, i. e.,

$$\begin{aligned}\dot{\boldsymbol{q}} \approx &\sum_{i=1}^{K}\frac{\sigma_i}{\sigma_i^2+\lambda^2}\boldsymbol{v}_i\,\boldsymbol{u}_i^{\mathrm{T}}\dot{\boldsymbol{t}}\\ &+\sum_{i=K+1}^{R}\frac{\sigma_i}{\sigma_i^2+\lambda^2+\beta^2}\boldsymbol{v}_i\,\boldsymbol{u}_i^{\mathrm{T}}\dot{\boldsymbol{t}}\,, \qquad (10.24)\end{aligned}$$

where the quality of the approximation is based on the accuracy of the estimates of $\tilde{\boldsymbol{u}}$ in (10.23). Notice that $K \leq R$; nevertheless, a nonzero value of λ is kept to guarantee satisfactory conditioning of the mapping (10.23) even for incorrect estimates of the output singular vectors.

Similarly, additional damping can be provided to the joint-space velocity components given by the in-

put singular vectors associated to the smallest singular values, since they are close to becoming the null-space velocities [10.22]. The solution can be generalized in the form

$$\dot{\boldsymbol{q}} = \left(\mathbf{J}_{\mathrm{t}}^{\mathrm{T}} \mathbf{J}_{\mathrm{t}} + \lambda^2 \mathbf{I} + \beta^2 \sum_{i=K+1}^{N} \tilde{\boldsymbol{v}}_i \tilde{\boldsymbol{v}}_i^{\mathrm{T}} \right)^{-1} \mathbf{J}_{\mathrm{t}}^{\mathrm{T}} \dot{\boldsymbol{t}} \,, \quad (10.25)$$

where β provides the largest contribution to damping only along the $N-K$ estimated null-space velocity components $\tilde{\boldsymbol{v}}_i$. Once again, this solution is an approximation to applying a greater selective filtering to the near singular components, i. e.,

$$\dot{\boldsymbol{q}} \approx \sum_{i=1}^{K} \frac{\sigma_i}{\sigma_i^2 + \lambda^2} \boldsymbol{v}_i \boldsymbol{u}_i^{\mathrm{T}} \dot{\boldsymbol{t}} + \sum_{i=K+1}^{R} \frac{\sigma_i}{\sigma_i^2 + \lambda^2 + \beta^2} \boldsymbol{v}_i \boldsymbol{u}_i^{\mathrm{T}} \dot{\boldsymbol{t}} \,, \quad (10.26)$$

where the quality of the approximation is based on the accuracy of the estimates $\tilde{\boldsymbol{v}}_i$ in (10.25). Also in this case a nonzero value of λ is kept to guarantee satisfactory conditioning of the mapping (10.25) even for incorrect estimates of the input singular vectors.

Comparison of (10.26) with (10.24) shows that, in the case of accurate estimate of the singular vectors, the solutions (10.23) and (10.25) behave identically.

10.3.3 Joint Trajectory Reconstruction

When solving the inverse kinematics problem at the first-order differential level, one obtains the joint velocity profile $\dot{\boldsymbol{q}}(t)$ that corresponds to an assigned end-effector task velocity profile $\dot{\boldsymbol{t}}(t)$; however, the robot motion controller needs reference position trajectories in addition to the velocity references for the joints. The reconstruction of the joint-position profile from the joint-velocity profile obtained from an end-effector task velocity profile $\dot{\boldsymbol{t}}(t)$ realizes a kinematic inversion that can be seen as an *inverse kinematics algorithm*.

If the joint-velocity profile were completely specified (e.g., by its analytical expression) the corresponding joint-position profile could be obtained from time integration

$$\boldsymbol{q}(t) = \boldsymbol{q}(t_0) + \int_{t_0}^{t} \dot{\boldsymbol{q}}(\boldsymbol{\tau}) \, \mathrm{d}\boldsymbol{\tau} \,. \quad (10.27)$$

However, digital implementation of the robot control system makes it more likely that a discrete-time sequence of samples $\dot{\boldsymbol{q}}_k$ of the computed joint velocities at the time instants t_k will be available, i. e.,

$$\dot{\boldsymbol{q}}_k = \dot{\boldsymbol{q}}(t_k) \,.$$

For this reason, a discrete-time numerical approximation of the continuous-time integral (10.27) must be suitably devised.

Accurate discrete-time approximations of a continuous-time integral usually require a trade-off between the complexity of the interpolating algorithm and the length of the time step. In real-time applications, such as robot motion control, high-order interpolation results in large finite-time delays that degrade the dynamic performance of the control loop. This time delay can be reduced by suitably shortening the time step; in fact, if the time step is sufficiently short even a low-order interpolation may give acceptable accuracy of the numerical integration. It is typical to use first-order interpolation, i. e., an Euler forward rectangular rule that transforms the integral (10.27) into

$$\boldsymbol{q}_k = \boldsymbol{q}_0 + \sum_{h=0}^{k-1} \dot{\boldsymbol{q}}_h \, \Delta t \,, \quad (10.28)$$

where Δt is the time step. Equation (10.28) is usually written in the more effective recursive form

$$\boldsymbol{q}_k = \boldsymbol{q}_{k-1} + \dot{\boldsymbol{q}}_{k-1} \, \Delta t \,.$$

Closed-Loop Inverse Kinematics (CLIK)

No matter what kind of interpolation is used, the small though unavoidable error suffered at each numerical integration step accumulates, leading to long-term drifting of the reconstructed profile from the exact joint-position profile. Another source of error affecting any integral reconstruction method is the possible uncertainty in the initial value of the joint position.

Algorithmic solutions that overcome these problems are based on the use of a feedback correction term; these are termed *closed-loop inverse kinematics* (CLIK) algorithms [10.23]. Considering, e.g., the case of first-order kinematics, the joint velocity at the k-th time instant is computed as (compare with (10.12))

$$\dot{\boldsymbol{q}}_k = \mathbf{J}_{\mathrm{t}}^{\dagger}(\boldsymbol{q}_k) \left\{ \dot{\boldsymbol{t}}_k + \mathbf{K} \left[\boldsymbol{t}_k - \boldsymbol{k}_{\mathrm{t}}(\boldsymbol{q}_k) \right] \right\} + \left[\mathbf{I} - \mathbf{J}_{\mathrm{t}}^{\dagger}(\boldsymbol{q}_k) \mathbf{J}_{\mathrm{t}}(\boldsymbol{q}_k) \right] \dot{\boldsymbol{q}}_{0k} \,, \quad (10.29)$$

where $\mathbf{K}$ is a constant positive-definite gain matrix.

Second-order CLIK algorithms are also available that solve for joint positions, velocities, and accelerations [10.24, 25]. The CLIK approach was originally proposed in [10.26] and [10.27] based on the Jacobian transpose in lieu of the pseudoinverse, which provides remarkable computational savings and provides inherent singularity handling capabilities [10.28].

10.4 Redundancy Resolution via Optimization

For a kinematically redundant manipulator, the inverse kinematics problem admits an infinite number of solutions, so that a criterion to select one of them is needed. In this section we consider the problem of redundancy resolution via optimization at the first-order differential kinematics level. Before discussing algorithmic schemes for computing joint velocities, we provide a short review of possible performance criteria.

10.4.1 Performance Criteria

The availability of degrees of freedom in excess of those strictly needed to execute a given task can be used to improve the value of performance criteria during the motion. These criteria may depend on the robot joint configuration only, or also involve velocities and/or accelerations.

Among the additional objectives that can be pursued by defining a suitable criterion, arguably the most popular is singularity avoidance. In fact, one of the original motivations for introducing kinematic redundancy was to increase the size of the workspace the manipulator could reach without being in a singular configuration (*unavoidable* singularities) for achieving general motion trajectory tracking tasks. A discussion of avoidable and unavoidable singularities in redundant manipulators can be found in [10.29]. If the assigned end-effector task does not pass through unavoidable singularities, it is in principle always possible to compute a joint trajectory along which the task Jacobian $\mathbf{J}_t$ is continuously full-rank. To this end, possible performance criteria are the configuration-dependent functions introduced in Sect. 10.2.2 that characterize the distance from singularities, i. e., the manipulability measure, the condition number, and the smallest singular value of $\mathbf{J}_t$. Maximizing (or keeping as large as possible) these functions during the motion is a reasonable plan in order to avoid singular configurations during the motion.

Because kinematic inversion can produce arbitrarily large joint velocities in the vicinity of singular configurations, a conceptually different possibility is to minimize the norm of the joint velocity generated by the redundancy resolution scheme. However, this approach guarantees singularity avoidance only if such a norm is minimized in an integral sense along the motion of the manipulator. Local minimization of the norm [10.10] does not lead to singularity avoidance in any practical sense [10.29].

Redundancy can be also used to keep the manipulator linkage away from undesired regions of the joint space, for example, mechanical joint limits that are typically present in robot manipulators may be avoided by minimizing the cost function [10.30]

$$\mathbf{H}(\boldsymbol{q}) = \frac{1}{2}\sum_{i=1}^{N}\left(\frac{q_i - q_{i,\mathrm{mid}}}{q_{i,\max} - q_{i,\min}}\right)^2 ,$$

where $[q_{i,\min}, q_{i,\max}]$ is the available range for joint i and $q_{i,\mathrm{mid}}$ is its midpoint. Another interesting application is obstacle avoidance, which can be enforced by minimizing suitable *artificial potential* functions defined on the basis of the image of the obstacle region in the configuration space [10.31, 32].

Many other performance criteria have been proposed in the literature; some of them are mentioned in Sects. 10.5 and 10.8.

10.4.2 Local Optimization

The simplest form of local optimization is represented by the pseudoinverse solution (10.13), which provides the joint velocity with the minimum norm among those that realize the task constraint. Clearly, the joint movement generated by this locally optimal solution does not provide global velocity minimization along the entire manipulator motion. This means that, despite the local minimization of joint velocities, singularity avoidance is not guaranteed [10.29].

Another possibility is to use the general solution (10.12), choosing the arbitrary joint velocity $\dot{\boldsymbol{q}}_0$ in the direction of the antigradient of a scalar configuration-dependent performance criteria $\mathbf{H}(\boldsymbol{q})$ that one would like to minimize

$$\dot{\boldsymbol{q}}_0 = -k_{\mathbf{H}}\,\nabla\mathbf{H}(\boldsymbol{q})\,, \tag{10.30}$$

where $k_{\mathbf{H}}$ is a scalar step size and $\nabla\mathbf{H}(\boldsymbol{q})$ denotes the gradient of $\mathbf{H}$ at the current joint configuration. This leads to the following redundancy resolution scheme [10.30]

$$\dot{\boldsymbol{q}} = \mathbf{J}_t^{\dagger}\dot{\boldsymbol{t}} - k_{\mathbf{H}}(\mathbf{I} - \mathbf{J}_t^{\dagger}\mathbf{J}_t)\,\nabla\mathbf{H}(\boldsymbol{q})\,. \tag{10.31}$$

Because its second term is the projection of the antigradient of $\mathbf{H}$ in the null space of the task Jacobian, the above expression is reminiscent of the projected gradient method for constrained minimization [10.33]. In particular, it can be shown [10.34] that the inverse kinematic solution (10.31) minimizes the complete quadratic function

$$L(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \frac{1}{2}\dot{\boldsymbol{q}}^{\mathrm{T}}\dot{\boldsymbol{q}} + k_{\mathbf{H}}\dot{\boldsymbol{q}}^{\mathrm{T}}\nabla\mathbf{H}(\boldsymbol{q})$$

at the current configuration $\boldsymbol{q}$. Thus, (10.31) represents a natural trade-off between the unconstrained local

minimization of the performance criteria $\mathbf{H}$ (which would lead to choose $\dot{\boldsymbol{q}} = -k_{\mathbf{H}} \nabla \mathbf{H}(\boldsymbol{q})$) and the satisfaction of constraint (10.2) by a minimum-norm joint velocity.

The choice of the step size $k_{\mathbf{H}}$ is critical for the performance of the redundancy resolution scheme (10.31). In particular, a small value of the step size may slow down the minimization of the performance criteria, but on the other hand a large value may even lead to an increase of $\mathbf{H}$ (recall that the antigradient is the *local* direction of maximum decrease). In practice, one may use a simplified *line search* technique such as Armijo's rule [10.33] to identify at each configuration an appropriate value of $k_{\mathbf{H}}$ in a reasonable time.

10.4.3 Global Optimization

The main advantage of the redundancy resolution scheme (10.31) is its simplicity: if the computation of $\nabla \mathbf{H}(\boldsymbol{q})$ and $k_{\mathbf{H}}$ is efficient, such a scheme is a realistic option for real-time kinematic inversion. Its disadvantage is to be found in the local nature of the optimization process, which can lead to unsatisfactory performance over long tasks, for example, use of (10.31) with $\mathbf{H} = -\mu$ (the manipulability measure) will perform better than the simple pseudoinverse solution, but still cannot guarantee singularity avoidance.

It is therefore natural to consider the possibility of selecting $\dot{\boldsymbol{q}}_0$ in (10.12) so as to minimize integral criteria of the form

$$\int_{t_\mathrm{i}}^{t_\mathrm{f}} \mathbf{H}(\boldsymbol{q}) \, \mathrm{d}t \, .$$

defined over the duration $[t_\mathrm{i}, t_\mathrm{f}]$ of the entire task (e.g., the integral of manipulability along the motion). Unfortunately, the solution of this problem (naturally formulated within the framework of calculus of variations) may not exist, and, in general, admits no closed form. One way to make the problem solvable is to include a quadratic form in the joint velocities or accelerations inside the integral. However, this is more easily done at the second-order kinematic level (Sect. 10.5).

10.5 Redundancy Resolution via Task Augmentation

Another approach to redundancy resolution consists of augmenting the task vector so as to tackle additional objectives expressed as constraints. In this section, we review the basic task augmentation techniques for solving the first-order differential kinematics (10.2).

10.5.1 Extended Jacobian

The extended Jacobian technique, proposed by *Baillieul* [10.35] and later revisited by *Chang* [10.36], enforces an appropriate number of functional constraints to be fulfilled along with the original end-effector task so as to identify a single solution among the infinite ones that are compatible with the end-effector task.

Let us consider an objective function $g(\boldsymbol{q})$ to be optimized and let also $\mathbf{N}_{\mathbf{J}_\mathrm{t}}(\boldsymbol{q})$ be a matrix spanning the null space of $\mathbf{J}_\mathrm{t}$ at a nonsingular configuration $\boldsymbol{q}$, e.g.,

$$\mathbf{N}_{\mathbf{J}_\mathrm{t}} = \mathbf{I} - \mathbf{J}_\mathrm{t}^\dagger \mathbf{J}_\mathrm{t} \, .$$

It can be recognized that, for a given $\boldsymbol{t}_0$, if $\boldsymbol{q}_0$ is a configuration at which the function $g(\boldsymbol{q})$ is at an extreme under the constraint $\boldsymbol{t}_0 = \boldsymbol{k}_\mathrm{t}(\boldsymbol{q}_0)$, one has

$$\left. \frac{\partial g(\boldsymbol{q})}{\partial \boldsymbol{q}} \right|_{\boldsymbol{q}=\boldsymbol{q}_0} \mathbf{N}_{\mathbf{J}_\mathrm{t}}(\boldsymbol{q}_0) = \mathbf{0}^\mathrm{T} \, . \tag{10.32}$$

If the Jacobian $\mathbf{J}_\mathrm{t}$ has full rank M, then $\mathbf{N}_{\mathbf{J}_\mathrm{t}}$ has rank $N-M$; therefore, (10.32) yields a set of $N-M$ independent constraints that can be written in vector form as

$$\boldsymbol{h}(\boldsymbol{q}) = \mathbf{0} \, ;$$

for example, these can be obtained by taking the scalar product of the gradient $\partial g(\boldsymbol{q})/\partial \boldsymbol{q}$ by each of the $N-M$ vectors constituting a base of the null space of $\mathbf{J}_\mathrm{t}$, i. e.,

$$\boldsymbol{h}(\boldsymbol{q}) = \left(\frac{\partial g(\boldsymbol{q})}{\partial \boldsymbol{q}} \; (\boldsymbol{v}_{M+1}(\boldsymbol{q}) \ldots \boldsymbol{v}_N(\boldsymbol{q})) \right)^\mathrm{T} \, .$$

At this point, the condition (10.32) implies that the equation

$$\begin{pmatrix} \boldsymbol{k}_\mathrm{t}(\boldsymbol{q}_0) \\ \boldsymbol{h}(\boldsymbol{q}_0) \end{pmatrix} = \begin{pmatrix} \boldsymbol{t}_0 \\ \mathbf{0} \end{pmatrix}$$

is satisfied.

For motion starting from $\boldsymbol{t}_0$ with $\boldsymbol{q}_0$ that tracks a trajectory $\boldsymbol{t}(t)$ by keeping $g(\boldsymbol{q})$ extremized at each time one has

$$\begin{pmatrix} \boldsymbol{k}_\mathrm{t}\left(\boldsymbol{q}(t)\right) \\ \boldsymbol{h}\left(\boldsymbol{q}(t)\right) \end{pmatrix} = \begin{pmatrix} \boldsymbol{t}(t) \\ \mathbf{0} \end{pmatrix} \, .$$

By differentiating both sides with respect to time one obtains

$$\begin{pmatrix} \mathbf{J}_t(\boldsymbol{q}) \\ \dfrac{\partial \boldsymbol{h}(\boldsymbol{q})}{\partial \boldsymbol{q}} \end{pmatrix} \dot{\boldsymbol{q}} = \begin{pmatrix} \dot{\boldsymbol{t}} \\ \boldsymbol{0} \end{pmatrix} , \tag{10.33}$$

where the matrix premultiplying the vector $\dot{\boldsymbol{q}}$ is square and is called the *extended Jacobian* $\mathbf{J}_{\text{ext}}$.

Therefore, if the initial configuration $\boldsymbol{q}_0$ extremizes $g(\boldsymbol{q})$ and provided that $\mathbf{J}_{\text{ext}}$ does not become singular, the time integral of the inverse mapping

$$\dot{\boldsymbol{q}} = \mathbf{J}_{\text{ext}}^{-1}(\boldsymbol{q}) \begin{pmatrix} \dot{\boldsymbol{t}} \\ 0 \end{pmatrix} \tag{10.34}$$

tracks the assigned end-effector trajectory $\dot{\boldsymbol{t}}(t)$ propagating joint configurations that extremize $g(\boldsymbol{q})$.

The extended Jacobian method has a major advantage over pseudoinverse techniques of the form (10.13) in that it is *cyclic* (Sect. 10.6). Moreover, solution (10.34) can be made equivalent to (10.12) via suitable choice of the vector $\dot{\boldsymbol{q}}_0$ [10.35, 37]. One relative disadvantage is its performance near algorithmic singularities (Sect. 10.5.3).

10.5.2 Augmented Jacobian

Another approach, the so-called task-space augmentation, introduces a constraint task to be fulfilled along with the end-effector task; then, an augmented Jacobian matrix is set-up whose inverse gives the sought joint velocity solution. The concept of task-space augmentation has been independently introduced by *Sciavicco* and *Siciliano* [10.28, 38, 39] and *Egeland* [10.40] and later revisited by *Seraji* in the framework of the configuration control method [10.41].

In detail, let us consider the vector $\boldsymbol{t}_c = (t_{c,1} \dots t_{c,P})^T$ that describes the tasks to be fulfilled in addition to the M-dimensional end-effector task $\boldsymbol{t}$. In the general case $P \leq N - M$, although full redundancy exploitation suggests the consideration of exactly as many additional tasks as the number of redundant degrees of freedom, i. e., $P = N - M$.

The relation between the joint-space coordinate vector $\boldsymbol{q}$ and the *constraint-task* vector $\boldsymbol{t}_c$ can be considered as a direct kinematics equation

$$\boldsymbol{t}_c = \boldsymbol{k}_c(\boldsymbol{q}) , \tag{10.35}$$

where $\boldsymbol{k}_c$ is a continuous nonlinear vector function. Accordingly, it is useful to consider the mapping

$$\dot{\boldsymbol{t}}_c = \mathbf{J}_c(\boldsymbol{q})\, \dot{\boldsymbol{q}} , \tag{10.36}$$

which can be obtained by differentiating Eq. (10.35). In (10.36), $\dot{\boldsymbol{t}}_c$ is the constraint-task velocity vector, and $\mathbf{J}_c(\boldsymbol{q}) = \partial \boldsymbol{k}_c / \partial \boldsymbol{q}$ is the $P \times N$ *constraint-task Jacobian* matrix.

At this point, an *augmented-task* vector can be defined by stacking the end-effector task vector with the constraint-task vector as

$$\boldsymbol{t}_a = \begin{pmatrix} \boldsymbol{t} \\ \boldsymbol{t}_c \end{pmatrix} = \begin{pmatrix} \boldsymbol{k}_t(\boldsymbol{q}) \\ \boldsymbol{k}_c(\boldsymbol{q}) \end{pmatrix} .$$

According to this definition, finding a joint configuration $\boldsymbol{q}$ that results in some desired value for $\boldsymbol{t}_a$ means satisfying both the end-effector and the constraint task at the same time.

A solution to this problem can be found at the differential level by inverting the mapping

$$\dot{\boldsymbol{t}}_a = \mathbf{J}_a(\boldsymbol{q})\, \dot{\boldsymbol{q}} , \tag{10.37}$$

where the matrix

$$\mathbf{J}_a = \begin{pmatrix} \mathbf{J}_t \\ \mathbf{J}_c \end{pmatrix}$$

is referred to as the *augmented Jacobian*.

A particular choice for the constraint-task vector is $\boldsymbol{t}_c = \boldsymbol{h}(\boldsymbol{q})$, with $\boldsymbol{h}$ defined as explained in 10.4.1, which allows the augmented Jacobian method to embed the extended Jacobian one.

10.5.3 Algorithmic Singularities

The specification of additional goals besides tracking the end-effector velocity raises the possibility that configurations exist at which the augmented kinematics problem is singular while the sole end-effector task kinematics is not; these configurations are referred to as *algorithmic singularities* [10.35]. In terms of the velocity mappings (10.33) and (10.37), an algorithmically singular configuration is one at which the extended and the augmented Jacobians, respectively, are singular while $\mathbf{J}_t$ is full-rank.

When he introduced task-augmentation redundancy resolution methods, *Baillieul* pointed out that algorithmic singularities are not a specific problem of the extended Jacobian technique but that they arise from the way in which the constraint task conflicts with the end-effector task [10.35, 37]. This is easily understandable in the simple problem of trajectory tracking with obstacle avoidance: if the assigned trajectory passes through an obstacle either the trajectory is tracked or the obstacle is avoided so that both tasks cannot be achieved. If the origin of the conflict between the two tasks has

a clear meaning the algorithmic singularity may then be avoided by keenly specifying the constraint task case by case [10.23]. In more general situations analytical tools may be useful in finding algorithmic singularities and guiding the choice of the constraint function [10.42] or in finding configurations that better harmonize the two tasks [10.43].

In terms of the tasks definitions given by (10.2) and (10.36), one can see that the two tasks are in conflict when

$$\mathcal{R}\left(\mathbf{J}_{\mathrm{c}}^{\mathrm{T}}\right) \cap \mathcal{R}\left(\mathbf{J}_{\mathrm{t}}^{\mathrm{T}}\right) \neq \{\mathbf{0}\}\,, \tag{10.38}$$

which is the condition for the occurrence of an algorithmic singularity. On the other hand, when

$$\mathcal{R}\left(\mathbf{J}_{\mathrm{c}}^{\mathrm{T}}\right) \cap \mathcal{R}\left(\mathbf{J}_{\mathrm{t}}^{\mathrm{T}}\right) = \{\mathbf{0}\}\,,$$

the two tasks are compatible becuse the two inverse mappings are linearly independent; a special case of task compatibility is when

$$\mathcal{R}\left(\mathbf{J}_{\mathrm{c}}^{\mathrm{T}}\right) \equiv \mathcal{R}^{\perp}\left(\mathbf{J}_{\mathrm{t}}^{\mathrm{T}}\right)\,, \tag{10.39}$$

and the two mappings are orthogonal.

At the algorithmic singularities the augmented Jacobian cannot be inverted but singularity-robust techniques can be adopted. Because the exact solution does not exist there are reconstruction errors and these affect both the task vectors. To counteract this problem the use of weighted damped least-squares has been considered to invert the augmented Jacobian matrix [10.44, 45]. A different approach is the so-called task priority inverse kinematics.

10.5.4 Task Priority

Conflicts between the end-effector task and the constraint task are handled in the framework of the task-priority strategy by suitably assigning an order of priority to the given tasks and then satisfying the lower-priority task only in the null space of the higher-priority task [10.46, 47]. In the typical case, an end-effector task is considered as the primary task, although examples can occur when it becomes the secondary task [10.23]. The idea is that, when an exact solution does not exist, the reconstruction error should only affect the lower-priority task.

With reference to solution (10.12), the task-priority method consists of computing $\dot{\boldsymbol{q}}_0$ to suitably achieve the P-dimensional constraint-task velocity $\dot{\boldsymbol{t}}_{\mathrm{c}}$. The projection of $\dot{\boldsymbol{q}}_0$ onto the null space of $\mathbf{J}_{\mathrm{t}}$ ensures that the lower priority of the constraint task will not affect the higher priority end-effector task [10.48].

When the secondary task $\dot{\boldsymbol{t}}_{\mathrm{c}}$ is orthogonal to the primary task $\dot{\boldsymbol{t}}$ (in the sense of (10.39)) the joint velocity

$$\dot{\boldsymbol{q}}_0 = \mathbf{J}_{\mathrm{c}}^{\dagger}(\boldsymbol{q})\dot{\boldsymbol{t}}_{\mathrm{c}} \tag{10.40}$$

would easily solve the problem, because it would already be a null-space velocity for the primary-task velocity mapping (10.2), i. e., projection through $\mathbf{N}_{\mathbf{J}_{\mathrm{t}}}$ would not be needed. However, in the general case the two tasks may be compatible but not orthogonal or may conflict and there may not exist a joint velocity solution that ensures the achievement of both $\dot{\boldsymbol{t}}$ and $\dot{\boldsymbol{t}}_{\mathrm{c}}$. To be consistent with the defined order of priority between the two tasks, a reasonable choice is then to guarantee exact tracking of the primary-task velocity while minimizing the constraint-task velocity reconstruction error $\dot{\boldsymbol{t}}_{\mathrm{c}} - \mathbf{J}_{\mathrm{c}}\dot{\boldsymbol{q}}$; this gives [10.49]

$$\dot{\boldsymbol{q}}_0 = \left[\mathbf{J}_{\mathrm{c}}(\mathbf{I} - \mathbf{J}_{\mathrm{t}}^{\dagger}\mathbf{J}_{\mathrm{t}})\right]^{\dagger}(\dot{\boldsymbol{t}}_{\mathrm{c}} - \mathbf{J}_{\mathrm{c}}\mathbf{J}_{\mathrm{t}}^{\dagger}\dot{\boldsymbol{t}})\,. \tag{10.41}$$

Finally, by observing that the null-space projection operator is both hermitian and idempotent, the solution given by (10.12) and (10.41) can be simplified to [10.46]

$$\dot{\boldsymbol{q}} = \mathbf{J}_{\mathrm{t}}^{\dagger}\dot{\boldsymbol{t}} + \left[\mathbf{J}_{\mathrm{c}}(\mathbf{I} - \mathbf{J}_{\mathrm{t}}^{\dagger}\mathbf{J}_{\mathrm{t}})\right]^{\dagger}(\dot{\boldsymbol{t}}_{\mathrm{c}} - \mathbf{J}_{\mathrm{c}}\mathbf{J}_{\mathrm{t}}^{\dagger}\dot{\boldsymbol{t}})\,. \tag{10.42}$$

It can be recognized that the problem of algorithmic singularities still remains; in fact, when condition (10.38) is satisfied the matrix $\mathbf{J}_{\mathrm{c}}(\mathbf{I} - \mathbf{J}_{\mathrm{t}}^{\dagger}\mathbf{J}_{\mathrm{t}})$ loses rank with full-rank $\mathbf{J}_{\mathrm{t}}$ and $\mathbf{J}_{\mathrm{c}}$. However, in contrast to the task-space augmentation approach, correct primary-task solutions are expected as long as the sole primary-task Jacobian matrix is full-rank. Away from the algorithmic singularities the task-priority strategy gives the same solution as the task-space augmentation approach; this implies that close to an algorithmic singularity the solution becomes ill-conditioned and large joint velocities may result. This problem can be solved by using a continuous version of the truncated SVD as described in [10.3].

Another approach is to relax minimization of the secondary-task velocity reconstruction constraint and simply pursue tracking of the components of (10.40) that do not conflict with the primary task [10.50, 51], namely

$$\dot{\boldsymbol{q}} = \mathbf{J}_{\mathrm{t}}^{\dagger}\dot{\boldsymbol{t}} + (\mathbf{I} - \mathbf{J}_{\mathrm{t}}^{\dagger}\mathbf{J}_{\mathrm{t}})\mathbf{J}_{\mathrm{c}}^{\dagger}\dot{\boldsymbol{t}}_{\mathrm{c}}\,. \tag{10.43}$$

An intuitive justification of this solution can be given as follows: The pseudoinverses $\mathbf{J}_t^\dagger$ and $\mathbf{J}_c^\dagger$ are used to solve separately for the joint velocities associated with the respective task velocities; the joint velocity associated with the (secondary) constraint task is then projected onto the null space of $\mathbf{J}_t$ to remove the components that would interfere with the (primary) end-effector task and finally added to the joint velocity associated with the end-effector task. As a result, a nice property of solution (10.43) is that algorithmic singularities are decoupled from the singularities of $\mathbf{J}_c$.

By construction, the solution (10.43) leads to larger constraint-task reconstruction errors than solution (10.42); this is the price to be paid for smoother trajectories for the joint velocity when tracking conflicting tasks. In any case, a CLIK implementation will allow one to recover from the secondary-task tracking error due to the algorithmic singularities. In this case, the CLIK formulation becomes

$$\dot{\boldsymbol{q}} = \mathbf{J}_t^\dagger \boldsymbol{w}_t + (\mathbf{I} - \mathbf{J}_t^\dagger \mathbf{J}_t)\mathbf{J}_c^\dagger \boldsymbol{w}_c \,,$$

with

$$\begin{aligned} \boldsymbol{w}_t &= \dot{\boldsymbol{t}} + \mathbf{K}_t \left(\boldsymbol{t} - \boldsymbol{k}_t(\boldsymbol{q})\right) \,, \\ \boldsymbol{w}_c &= \dot{\boldsymbol{t}}_c + \mathbf{K}_c \left(\boldsymbol{t}_c - \boldsymbol{k}_c(\boldsymbol{q})\right) \,. \end{aligned}$$

10.6 Second-Order Redundancy Resolution

Redundancy resolution at the acceleration level allows the consideration of dynamic performance along the manipulator motion. Moreover, the obtained acceleration profiles can be directly used as reference signals (together with the corresponding positions and velocities) of a task-space dynamic controller. On the other hand, a second-order redundancy resolution scheme is invariably more demanding in terms of computational load.

The simplest scheme operating at the acceleration level is represented by (10.15), i. e., the solution with the minimum norm among those that realize the task constraint (10.8). Similar to the velocity-level pseudoinverse solution, the joint movement generated by this locally optimal solution does not result in global acceleration minimization along the entire manipulator motion. Remarkably, however, the use of (10.15) does lead to the minimization of the integral index

$$\int_{t_i}^{t_f} \dot{\boldsymbol{q}}^T \dot{\boldsymbol{q}} \, \mathrm{d}t \,,$$

over $[t_i, t_f]$ provided that the appropriate boundary conditions are satisfied [10.52]. For example, in the case of free endpoints (neither joint positions nor velocities specified at t_i and t_f) the boundary conditions to be satisfied are split and expressed as

$$\dot{\boldsymbol{q}}(\bar{t}) = \mathbf{J}_t^\dagger \dot{\boldsymbol{t}}(\bar{t}) \,, \quad \bar{t} = t_i, t_f \,.$$

Thus, in spite of the apparent simplicity and elegance of the solution (10.15), actual minimization of the above integral requires the solution of a two-point boundary value problem (TPBVP), which is a computationally intensive numerical procedure impractical for real-time kinematic control. However, this may be perfectly acceptable for offline redundancy resolution in an industrial setting.

More flexibility in the choice of (both local and global) performance criteria is obviously obtained by considering the full second-order solution (10.14). Let the manipulator dynamic model be expressed as

$$\boldsymbol{\tau} = \mathbf{H}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{c}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \boldsymbol{\tau}_g(\boldsymbol{q}) \,, \tag{10.44}$$

where $\boldsymbol{\tau}$ is the vector of actuator torques, $\mathbf{H}$ is the manipulator inertia matrix, $\boldsymbol{c}$ is the vector of centrifugal/Coriolis terms, and $\boldsymbol{\tau}_g$ is the gravitational torque vector. Choosing the null-space acceleration in (10.14) as

$$\ddot{\boldsymbol{q}}_0 = -\left[\mathbf{H}(\mathbf{I} - \mathbf{J}_t^\dagger \mathbf{J}_t)\right]^\dagger \tilde{\boldsymbol{\tau}} \,, \tag{10.45}$$

with

$$\tilde{\boldsymbol{\tau}} = \mathbf{H}\mathbf{J}_t^\dagger (\ddot{\boldsymbol{t}} - \dot{\mathbf{J}}_t \dot{\boldsymbol{q}}) + \boldsymbol{c} + \boldsymbol{\tau}_g$$

leads to the local minimization of the actuator torque norm $\boldsymbol{\tau}^T\boldsymbol{\tau}$ [10.53]. This particular redundancy resolution scheme performs reasonably over short tasks, but may lead to instability (more precisely, very high joint torques) in the long run, essentially due to the build-up of null-space joint velocities. Note also that the matrix product $\mathbf{H}(\mathbf{I} - \mathbf{J}_t^\dagger \mathbf{J}_t)$ in (10.45) is not full-rank; however, its pseudoinverse can be efficiently calculated using the procedure given in [10.54]. Minimization of the integral joint torque is also possible [10.55]; this solution obviously avoids the instability problem, but once again the solution of a TPBVP is required.

Another interesting inverse solution is the following

$$\ddot{\boldsymbol{q}} = \mathbf{J}_{\mathrm{t},\mathbf{H}}^{\dagger}(\ddot{\boldsymbol{t}} - \dot{\mathbf{J}}_{\mathrm{t}}\,\dot{\boldsymbol{q}}) + (\mathbf{I} - \mathbf{J}_{\mathrm{t},\mathbf{H}}^{\dagger}\mathbf{J}_{\mathrm{t}})\,\mathbf{H}^{-1}\boldsymbol{c}\,, \qquad (10.46)$$

which is slightly modified with respect to the general formula (10.14) in view of the use of a weighted pseudoinverse. In particular, $\mathbf{J}_{\mathrm{t},\mathbf{H}}^{\dagger}$ is the inertia-weighted task Jacobian pseudoinverse, given in the full-rank case by the following expression

$$\mathbf{J}_{\mathrm{t},\mathbf{H}}^{\dagger} = \mathbf{H}^{-1}\mathbf{J}_{\mathrm{t}}^{\mathrm{T}}(\mathbf{J}_{\mathrm{t}}\,\mathbf{H}^{-1}\mathbf{J}_{\mathrm{t}}^{\mathrm{T}})^{-1}\,.$$

The solution (10.46) minimizes

$$\int_{t_{\mathrm{i}}}^{t_{\mathrm{f}}} \frac{1}{2}\,\dot{\boldsymbol{q}}^{\mathrm{T}}\mathbf{H}(\boldsymbol{q})\dot{\boldsymbol{q}}\;\mathrm{d}t\,,$$

i. e., the integral over $[t_{\mathrm{i}}, t_{\mathrm{f}}]$ of the manipulator kinetic energy [10.52]. Once again, the correct boundary conditions must be used; for example, the case of free endpoints leads to

$$\dot{\boldsymbol{q}}(\bar{t}) = \mathbf{J}_{\mathrm{t},\mathbf{H}}^{\dagger}\dot{\boldsymbol{t}}(\bar{t})\,, \quad \bar{t} = t_{\mathrm{i}}, t_{\mathrm{f}}\,.$$

10.7 Cyclicity

A common drawback of redundancy resolution schemes based on differential kinematics is the lack of *cyclicity* (also called *repeatability*): in general, the joint-space trajectory corresponding to a cyclic task space trajectory is not cyclic itself (i. e., the final position of the joints does not coincide with the initial position). This phenomenon is clearly undesirable, because it basically means that the behavior of the manipulator along periodic tasks to be repeated over and over is unpredictable.

A mathematical condition for cyclicity exists for a particular class of redundancy resolution methods [10.56]. In particular, consider any scheme of the form

$$\dot{\boldsymbol{q}} = \mathbf{G}_{\mathrm{t}}(\boldsymbol{q})\dot{\boldsymbol{t}}\,, \qquad (10.47)$$

where $\mathbf{G}_{\mathrm{t}}$ is any *generalized inverse* of the task Jacobian $\mathbf{J}_{\mathrm{t}}$, i. e., an $N \times M$ matrix such that $\mathbf{J}_{\mathrm{t}}\mathbf{G}_{\mathrm{t}}\mathbf{J}_{\mathrm{t}} = \mathbf{J}_{\mathrm{t}}$ (the pseudoinverse matrix $\mathbf{J}_{\mathrm{t}}^{\dagger}$ in (10.13) is a particular generalized inverse). Assume that the assigned task $\boldsymbol{t}(t)$ describes a cyclic trajectory in a simply connected region of the task space, and denote by $\boldsymbol{g}_{\mathrm{t}i}(\boldsymbol{q})$ the i-th column of $\mathbf{G}_{\mathrm{t}}$. A necessary and sufficient condition for (10.47) to generate cyclic joint trajectories is that the *distribution*

$$\Delta_{\mathbf{G}_{\mathrm{t}}}(\boldsymbol{q}) = \mathrm{span}\{\boldsymbol{g}_{\mathrm{t}1}(\boldsymbol{q}), \ldots, \boldsymbol{g}_{\mathrm{t}M}(\boldsymbol{q})\}$$

is involutive (i. e., it is closed under the Lie bracket operation).

It should be emphasized that the involutivity of $\Delta_{\mathbf{G}_{\mathrm{t}}}$ is a strong condition, because it must be satisfied at any configuration. This suggests that most generalized inverses are not cyclic. Note also that the above condition for cyclicity depends on the chosen generalized inverse (i. e., pseudoinverse, weighted pseudoinverse, and so on) as well as on the form of $\mathbf{J}_{\mathrm{t}}$, which in turn is related to the mechanical structure of the manipulator. This means that cyclicity must be established on a case-by-case basis. However, it is possible to design a repeatable inverse that can approximate any desired generalized inverse over a specified subset of the workspace [10.57].

As for redundancy resolution schemes which do not fall in the class (10.47) – namely, those entailed by the general inverse solution (10.12) – they are in general not cyclic. In particular, this is true for when local optimization is used to solve redundancy, as in (10.31). A notable exception is the extended Jacobian method, which is always cyclic.

10.8 Fault Tolerance

One natural use for kinematic redundancy is that of fault tolerance, i. e., if there are more degrees of freedom than the minimum required to complete a task then one should, theoretically, be able to tolerate a joint failure and still complete the desired task. While component failures for robots employed in structured and benign environments where regular maintenance can be performed are relatively rare, there are many important applications, although less common, where this is not true, e.g., in space or underwater exploration and in nuclear environments. The failure rates for components in such harsh environments are relatively high [10.58] and maintenance is not possible. Many of these component failures will result in a robot's joint becoming immobilized, i. e., a locked joint failure mode [10.59]. Component failures that result in other common fail-

Avidbots Ex1016
Page 281 of 2154

ure modes, e.g., free-swinging joint failures [10.60], are also frequently transformed into the locked joint failure mode by failure recovery mechanisms that employ fail safe brakes [10.61].

A large body of work on fault-tolerant manipulators has focused on the kinematic properties of redundant robots, both of serial and parallel structure [10.62]. These analyses have been performed both on the local properties associated with the manipulator Jacobian [10.63, 64] as well as the global characteristics such as the resulting workspace following a particular failure [10.65–67]. Clearly both local and global kinematic properties are related, e.g., if a robot is at a workspace boundary then the Jacobian is singular.

The effect of locked joint failures on the motion capabilities of a robotic manipulator are most easily characterized when the manipulator Jacobian is represented as a collection of columns

$$\mathbf{J}_{M\times N} = \begin{pmatrix} \mathcal{J}_1 & \mathcal{J}_2 & \dots & \mathcal{J}_N \end{pmatrix} ,$$

where $\mathcal{J}_i$ represents the end-effector velocity due to the velocity of joint i. For an arbitrary single joint failure at joint f, assuming that the failed joint can be locked, the resulting $(M\times(N-1))$ Jacobian will be missing the f-th column, where f can range from 1 to N. This Jacobian will be denoted by a preceding superscript so that in general

$$^f\mathbf{J}_{M\times(N-1)} = \begin{pmatrix} \mathcal{J}_1 & \mathcal{J}_2 & \dots & \mathcal{J}_{f-1} & \mathcal{J}_{f+1} & \dots & \mathcal{J}_N \end{pmatrix}.$$

As noted above, the local velocity properties of a manipulator Jacobian are frequently quantified in terms of the singular values. Most local dexterity measures can be defined in terms of simple combinations of these singular values such as their product (determinant), sum (trace), or ratio (condition number). The most significant of the singular values is σ_M, the minimum singular value, because it is by definition the measure of proximity to a singularity and tends to dominate the behavior of both the manipulability (determinant) and the condition number. The minimum singular value is also a measure of the worst-case dexterity over all possible end-effector motions. One definition of failure tolerance is based on the worst-case dexterity following an arbitrary locked joint failure. Because $^f\sigma_M$ denotes the minimum singular value of $^f\mathbf{J}$, it is a measure of the worst-case dexterity if joint f fails. If all joints are equally likely to fail, then a measure of the worst-case failure tolerance is given by

$$K = \min_{f=1}^{N} {}^f\sigma_M \, .$$

Physically, this amounts to minimizing the worst-case increase in joint velocity when a joint is locked and the others must accelerate to maintain the desired end effector trajectory. In addition, maximizing K is equivalent to locally maximizing the distance to a post-failure workspace boundary [10.68]. To insure that manipulator performance is optimal prior to a failure, one may want to further define an optimally failure tolerant Jacobian as having all equal singular values due to the desirable properties of isotropic manipulator configurations. Under these conditions, to guarantee that the minimum $^f\sigma_M$ is as large as possible they should all be equal. It is easy to show that the worst-case dexterity of an isotropic manipulator that experiences a single joint failure is governed by the inequality

$$K = \min_{f=1}^{N} {}^f\sigma_M \leq \sigma\sqrt{\frac{(N-M)}{N}} \, ,$$

where σ denotes the norm of the original Jacobian. The best case of equality occurs if the manipulator is in an optimally failure tolerant configuration. The above inequality makes sense from a physical point of view because it represents the ratio of the degree of redundancy to the original number of degrees of freedom. Using the above definition of an optimally failure tolerant configuration one can identify the structure of the Jacobian required to obtain this property [10.63]. In particular, one can show that the optimally failure tolerant criteria requires that each joint contributes equally to the null space of the Jacobian transformation. Physically, this means that the redundancy of the robot is uniformly distributed among all the joints so that a fail-

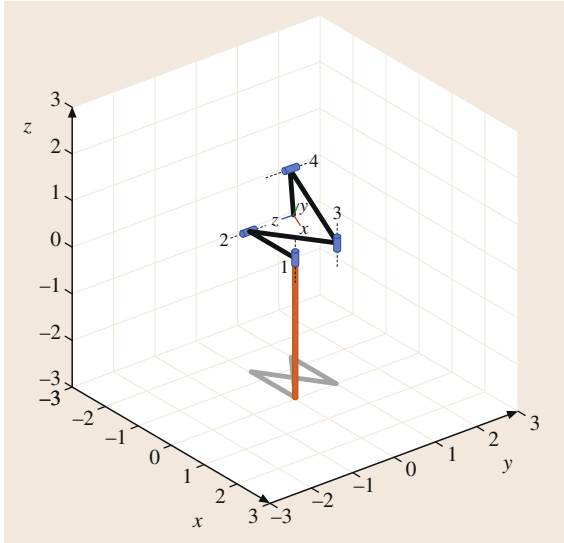


Fig. 10.9 A four degree-of-freedom spatial positioning manipulator in a configuration that is optimally fault tolerant

ure at any one joint can be compensated for by the remaining joints.

A simple example of an optimally failure tolerant configuration that is easy to visualize is that of a spatial positioning manipulator with four joints whose Jacobian is given by

$$\mathbf{J}_{\mathrm{v}} = \begin{pmatrix} -\sqrt{3/4} & \sqrt{1/12} & \sqrt{1/12} & \sqrt{1/12} \\ 0 & -\sqrt{2/3} & \sqrt{1/6} & \sqrt{1/6} \\ 0 & 0 & -\sqrt{1/2} & \sqrt{1/2} \end{pmatrix} ,$$

where $\mathbf{J}_{\mathrm{v}}$ represents the linear velocity portion of a manipulator Jacobian. The null space at this configuration is given by $(1/2)[1\,1\,1\,1]^{\mathrm{T}}$, which illustrates that each joint contributes equally to the null space motion, thus distributing the redundancy proportionally to all degrees of freedom. If the four possible single locked joint failures are considered, one can show that

$$^{f}\sigma_3 = K_{\max} = \frac{1}{2} ,$$
$$f = 1, \ldots, 4 ,$$

which satisfies the optimally failure tolerant criterion. It is then possible to design physical robot manipulators that possess this Jacobian along with its associated optimal kinematic properties. An example of one such robot is shown in Fig. 10.9 [10.69].

10.9 Conclusion and Further Reading

Research on kinematically redundant robots has been flourishing for over two decades now, and is still very active. The number of papers dealing with this subject is therefore enormous. The few works cited below are simply a small addition to the fundamental contributions already referenced so far, and constitute by no means an exhaustive list.

The mechanical design of kinematically redundant manipulators has been studied in many papers [10.70–74]. In particular, the superiority of human-arm-like manipulators over conventional 6-DOF robots was first advocated in [10.75]. Reconfiguration of this arm so as to ensure full mobility in the whole workspace is studied in [10.76, 77].

A general analysis of the inverse kinematic problem for redundant manipulators is presented in [10.78]. In particular, the geometric structure of self-motions is analyzed in [10.79].

Weighting the damped least-squares solution for guaranteed singularity avoidance in anisotropic end-effector tasks was proposed in [10.9, 21, 80].

In addition to singularity avoidance, redundancy has also been exploited to achieve obstacle avoidance [10.37, 46, 81, 82], minimization of the effects of joint elasticity [10.83], fault tolerance [10.60], reduction of impact force [10.84, 85], and maximization of various dexterity measures [10.4, 5, 43, 86]. A completely different approach for obstacle avoidance with redundant robots is presented in [10.87].

A review of redundancy resolution via local optimization is given in [10.88]. A numerically efficient alternative for redundancy resolution via local optimization is proposed in [10.89]. A redundancy resolution approach with somehow intermediate characteristics between local and global optimization is presented in [10.90]. Other methods for second-order redundancy resolution are discussed, e.g., in [10.91]. Also worth citing is the dynamically consistent generalized inverse of [10.92].

The cyclicity issue was first pointed out in [10.93], and further investigated, e.g., in [10.94, 95]. General formalisms for redundancy resolution of vehicle-manipulator systems subject to nonholonomic constraints are described in [10.96, 97].

Video-References

VIDEO 813 KUKA LBR iiwa – Kinematic redundancy
available from http://handbookofrobotics.org/view-chapter/10/videodetails/813

VIDEO 814 Free floating autonomous valve turning (task priority redundancy control + task concurrence)
available from http://handbookofrobotics.org/view-chapter/10/videodetails/814

VIDEO 815 Human inspired tele-impedance and minimum effort controller for improved manipulation performance
available from http://handbookofrobotics.org/view-chapter/10/videodetails/815

VIDEO 816 Human motion mapping to a robot arm with redundancy resolution
available from http://handbookofrobotics.org/view-chapter/10/videodetails/816

VIDEO 817 Configuration space control of KUKA lightweight robot LWR with EXARM exoskeleton
available from http://handbookofrobotics.org/view-chapter/10/videodetails/817

VIDEO 818 FlexIRob – Teaching nullspace constraints in physical human-robot interaction
available from http://handbookofrobotics.org/view-chapter/10/videodetails/818

VIDEO 819 Visual servoing control of baxter robot arms with obstacle avoidance using kinematic redundancy
available from http://handbookofrobotics.org/view-chapter/10/videodetails/819

References

10.1 D.E. Whitney: Resolved motion rate control of manipulators and human prostheses, IEEE Trans. Man-Mach. Syst. **10**(2), 47–53 (1969)

10.2 G.H. Golub, C. Reinsch: Singular value decomposition and least-squares solutions, Numer. Math. **14**, 403–420 (1970)

10.3 A.A. Maciejewski, C.A. Klein: The singular value decomposition: computation and applications to robotics, Int. J. Robotics Res. **8**(6), 63–79 (1989)

10.4 T. Yoshikawa: Manipulability of robotic mechanisms, Int. J. Robotics Res. **4**(2), 3–9 (1985)

10.5 C.A. Klein, B.E. Blaho: Dexterity measures for the design and control of kinematically redundant manipulators, Int. J. Robotics Res. **6**(2), 72–83 (1987)

10.6 A.A. Maciejewski, C.A. Klein: Numerical filtering for the operation of robotic manipulators through kinematically singular configurations, J. Robotics Syst. **5**, 527–552 (1988)

10.7 S. Chiaverini: Estimate of the two smallest singular values of the Jacobian matrix: application to damped least-squares inverse kinematics, J. Robotics Syst. **10**, 991–1008 (1993)

10.8 O. Egeland, M. Ebdrup, S. Chiaverini: Sensory control in singular configurations – Application to visual servoing, IEEE Int. Workshop Intell. Motion Control, Istanbul (1990) pp. 401–405

10.9 Y. Nakamura, H. Hanafusa: Inverse kinematic solutions with singularity robustness for robot manipulator control, Trans. ASME J. Dyn. Syst. Meas., Control **108**, 163–171 (1986)

10.10 D.E. Whitney: The mathematics of coordinated control of prosthetic arms and manipulators, Trans. ASME J. Dyn. Syst. Meas., Control **94**, 303–309 (1972)

10.11 T.L. Boullion, P.L. Odell: *Generalized Inverse Matrices* (Wiley, New York 1971)

10.12 C.R. Rao, S.K. Mitra: *Generalized Inverse of Matrices and its Applications* (Wiley, New York 1971)

10.13 A. Ben-Israel, T.N.E. Greville: *Generalized Inverses: Theory and Applications* (Wiley, New York 1974)

10.14 R.H. Taylor: Planning and execution of straight-line manipulator trajectories, IBM J. Res. Dev. **23**, 424–436 (1979)

10.15 M. Sampei, K. Furuta: Robot control in the neighborhood of singular points, IEEE J. Robotics Autom. **4**, 303–309 (1988)

10.16 E.W. Aboaf, R.P. Paul: Living with the singularity of robot wrists, IEEE Int. Conf. Robotics Autom. (ICRA), Raleigh (1987) pp. 1713–1717

10.17 S. Chiaverini, O. Egeland: A solution to the singularity problem for six-joint manipulators, IEEE Int. Conf. Robotics Autom. (ICRA), Cincinnati (1990) pp. 644–649

10.18 S. Chiaverini, O. Egeland: An efficient pseudo-inverse solution to the inverse kinematic problem for six-joint manipulators, Model. Identif. Control **11**(4), 201–222 (1990)

10.19 O. Khatib: A unified approach for motion and force control of robot manipulators: The operational space formulation, IEEE J. Robotics Autom. **3**, 43–53 (1987)

10.20 C.W. Wampler: Manipulator inverse kinematic solutions based on vector formulations and damped least-squares methods, IEEE Trans. Syst. Man Cybern. **16**, 93–101 (1986)

10.21 S. Chiaverini, O. Egeland, R.K. Kanestrøm: Achieving user-defined accuracy with damped least-squares inverse kinematics, 5th Int. Conf. Adv. Robotics, Pisa (1991) pp. 672–677

10.22 J.R. Sagli: Coordination of Motion in Manipulators with Redundant Degrees of Freedom, Ph.D. Thesis (Norwegian University of Science and Technology, Trondheim 1991)

10.23 P. Chiacchio, S. Chiaverini, L. Sciavicco, B. Siciliano: Closed-loop inverse kinematics schemes for constrained redundant manipulators with task space augmentation and task priority strategy, Int. J. Robotics Res. **10**(4), 410–425 (1991)

10.24 B. Siciliano: A closed-loop inverse kinematic scheme for on-line joint-based robot control, Robotica **8**, 231–243 (1990)

10.25 Z.R. Novaković, B. Siciliano: A new second-order inverse kinematics solution for redundant manipulators. In: *Advances in Robot Kinematics*, ed. by S. Stifter, J. Lenarčič (Springer, New York 1991) pp. 408–415

10.26 A. Balestrino, G. De Maria, L. Sciavicco: Robust control of robotic manipulators, 9th IFAC World Cong., Budapest (1984) pp. 80–85

10.27 W.A. Wolovich, H. Elliott: A computational technique for inverse kinematics, 23rd IEEE Conf. Decis. Control, Las Vegas (1984) pp. 1359–1363

10.28 L. Sciavicco, B. Siciliano: A solution algorithm to the inverse kinematic problem for redundant manipulators, IEEE J. Robotics Autom. **4**, 403–410 (1988)

10.29 J. Baillieul, J. Hollerbach, R.W. Brockett: Programming and control of kinematically redundant manipulators, 23rd IEEE Conf. Decis. Control, Las Vegas (1984) pp. 768–774

10.30 A. Liégeois: Automatic supervisory control of the configuration and behavior of multibody mech-

anisms, IEEE Trans. Syst. Man Cybern. **7**, 868–871 (1977)
10.31 O. Khatib: Real-time obstacle avoidance for manipulators and mobile robots, IEEE Int. Conf. Robotics Autom. (ICRA), St. Louis (1985) pp. 500–505
10.32 J.C. Latombe: *Robot Motion Planning* (Kluwer, Boston 1991)
10.33 D.G. Luenberger: *Linear and Nonlinear Programming* (Addison-Wesley, Reading 1984)
10.34 A. De Luca, G. Oriolo: Issues in acceleration resolution of robot redundancy, 3rd IFAC Symp. Robot Control, Vienna (1991) pp. 665–670
10.35 J. Baillieul: Kinematic programming alternatives for redundant manipulators, IEEE Int. Conf. Robotics Autom. (ICRA), St. Louis (1985) pp. 722–728
10.36 P.H. Chang: A closed-form solution for inverse kinematics of robot manipulators with redundancy, IEEE J. Robotics Autom. **3**, 393–403 (1987)
10.37 J. Baillieul: Avoiding obstacles and resolving kinematic redundancy, IEEE Int. Conf. Robotics Autom. (ICRA), San Francisco (1986) pp. 1698–1704
10.38 L. Sciavicco, B. Siciliano: Solving the inverse kinematic problem for robotic manipulators, 6th CISM-IFToMM Symp. Theory Pract. Robotics Manip., Kraków (1986) pp. 107–114
10.39 L. Sciavicco, B. Siciliano: A dynamic solution to the inverse kinematic problem for redundant manipulators, IEEE Int. Conf. Robotics Autom. (ICRA), Raleigh (1987) pp. 1081–1087
10.40 O. Egeland: Task-space tracking with redundant manipulators, IEEE J. Robotics Autom. **3**, 471–475 (1987)
10.41 H. Seraji: Configuration control of redundant manipulators: theory and implementation, IEEE J. Robotics Autom. **5**, 472–490 (1989)
10.42 J. Baillieul: A constraint oriented approach to inverse problems for kinematically redundant manipulators, IEEE Int. Conf. Robotics Autom. (ICRA), Raleigh (1987) pp. 1827–1833
10.43 S.L. Chiu: Control of redundant manipulators for task compatibility, IEEE Int. Conf. Robotics Autom. (ICRA), Raleigh (1987) pp. 1718–1724
10.44 H. Seraji, R. Colbaugh: Improved configuration control for redundant robots, J. Robotics Syst. **7**, 897–928 (1990)
10.45 O. Egeland, J.R. Sagli, I. Spangelo, S. Chiaverini: A damped least-squares solution to redundancy resolution, IEEE Int. Conf. Robotics Autom. (ICRA), Sacramento (1991) pp. 945–950
10.46 A.A. Maciejewski, C.A. Klein: Obstacle avoidance for kinematically redundant manipulators in dynamically varying environments, Int. J. Robotics Res. **4**(3), 109–117 (1985)
10.47 Y. Nakamura, H. Hanafusa, T. Yoshikawa: Task-priority based redundancy control of robot manipulators, Int. J. Robotics Res. **6**(2), 3–15 (1987)
10.48 H. Hanafusa, T. Yoshikawa, Y. Nakamura: Analysis and control of articulated robot arms with redundancy, IFAC 8th Trienn. World Congr., Kyoto (1981) pp. 78–83
10.49 Y. Nakamura, H. Hanafusa: Task priority based redundancy control of robot manipulators. In: *Robotics Research – The Second International Symposium*, ed. by H. Hanafusa, H. Hinoue (MIT Press, Cambridge 1985) pp. 155–162
10.50 S. Chiaverini: Task-priority redundancy resolution with robustness to algorithmic singularities, 4th IFAC Symp. Robot Control, Capri (1994) pp. 393–399
10.51 S. Chiaverini: Singularity-robust task-priority redundancy resolution for real-time kinematic control of robot manipulators, IEEE Trans. Robotics Autom. **13**, 398–410 (1997)
10.52 K. Kazerounian, Z. Wang: Global versus local optimization in redundancy resolution of robotic manipulators, Int. J. Robotics Res. **7**(5), 312 (1988)
10.53 J.M. Hollerbach, K.C. Suh: Redundancy resolution of manipulators through torque optimization, IEEE J. Robotics Autom. **3**, 308–316 (1987)
10.54 A.A. Maciejewski: Kinetic limitations on the use of redundancy in robotic manipulators, IEEE Trans. Robotics Autom. **7**(2), 205–210 (1991)
10.55 J.M. Hollerbach, K.C. Suh: Local versus global torque optimization of redundant manipulators, IEEE Int. Conf. Robotics Autom. (ICRA), Raleigh (1987) pp. 619–624
10.56 T. Shamir, Y. Yomdin: Repeatability of redundant manipulators: Mathematical solution of the problem, IEEE Trans. Autom. Control **33**, 1004–1009 (1988)
10.57 R.G. Roberts, A.A. Maciejewski: Repeatable generalized inverse control strategies for kinematically redundant manipulators, IEEE Trans. Autom. Control **38**(5), 689–699 (1993)
10.58 B.S. Dhillon, A.R.M. Fashandi, K.L. Liu: Robot systems reliability and safety: A review, J. Qual. Maint. Eng. **8**(3), 170–212 (2002)
10.59 M.L. Visinsky, J.R. Cavallaro, I.D. Walker: A dynamic fault tolerance framework for remote robots, IEEE Trans. Robotics Autom. **11**(4), 477–490 (1995)
10.60 J.D. English, A.A. Maciejewski: Fault tolerance for kinematically redundant manipulators: Anticipating free-swinging joint failures, IEEE Trans. Robotics Autom. **14**, 566–575 (1998)
10.61 P. Nieminen, S. Esque, A. Muhammad, J. Mattila, J. Väyrynen, M. Siuko, M. Vilenius: Water hydraulic manipulator for fail safe and fault tolerant remote handling operations at ITER, Fusion Eng. Des. **84**(7), 1420–1424 (2009)
10.62 Y. Yi, J.E. McInroy, Y. Chen: Fault tolerance of parallel manipulators using task space and kinematic redundancy, IEEE Trans. Robotics **22**(5), 1017–1021 (2006)
10.63 A.A. Maciejewski, R.G. Roberts: On the existence of an optimally failure tolerant 7R manipulator Jacobian, Appl. Math. Comp. Sci. **5**(2), 343–357 (1995)
10.64 R.G. Roberts, A.A. Maciejewski: A local measure of fault tolerance for kinematically redundant manipulators, IEEE Trans. Robotics Autom. **12**(4), 543–552 (1996)
10.65 C.J.J. Paredis, P.K. Khosla: Fault tolerant task execution through global trajectory planning, Rel. Eng. Syst. Saf. **53**, 225–235 (1996)

Avidbots Ex1016
Page 285 of 2154

10.66 C.L. Lewis, A.A. Maciejewski: Fault tolerant operation of kinematically redundant manipulators for locked joint failures, IEEE Trans. Robotics Autom. **13**(4), 622–629 (1997)

10.67 R.S. Jamisola Jr., A.A. Maciejewski, R.G. Roberts: Failure tolerant path planning for kinematically redundant manipulators anticipating locked-joint failures, IEEE Trans. Robotics **22**(4), 603–612 (2006)

10.68 K.N. Groom, A.A. Maciejewski, V. Balakrishnan: Real-time failure-tolerant control of kinematically redundant manipulators, IEEE Trans. Robotics Autom. **15**(6), 1109–1116 (1999)

10.69 K.M. Ben-Gharbia, A.A. Maciejewski, R.G. Roberts: Kinematic design of redundant robotic manipulators for spatial positioning that are optimally fault tolerant, IEEE Trans. Robotics Autom. **29**(5), 1300–1307 (2013)

10.70 J. Salisbury, J. Abramowitz: Design and control of a redundant mechanism for small motion, IEEE Int. Conf. Robotics Autom. (ICRA), St. Louis (1985) pp. 323–328

10.71 J. Baillieul: Design of kinematically redundant mechanisms, 24th IEEE Conf. Decis. Control, Ft. Lauderdale (1985) pp. 18–21

10.72 G.S. Chirikjian, J.W. Burdick: Design and experiments with a 30 DoF robot, IEEE Int. Conf. Robotics Autom. (ICRA), Atlanta (1993) pp. 113–119

10.73 J. Angeles: The design of isotropic manipulator architectures in the presence of redundancies, Int. J. Robotics Res. **11**(3), 196–201 (1992)

10.74 A. Bowling, O. Khatib: Design of macro/mini manipulators for optimal dynamic performance, IEEE Int. Conf. Robotics Autom. (ICRA), Albuquerqe (1997) pp. 449–454

10.75 J.M. Hollerbach: Optimum kinematic design for a seven degree of freedom manipulator. In: *Robotics Research – The Second International Symposium*, ed. by H. Hanafusa, H. Hinoue (MIT Press, Cambridge 1985) pp. 216–222

10.76 O. Egeland, J.R. Sagli, S. Hendseth, F. Wilhelmsen: Dynamic coordination in a manipulator with seven joints, IEEE Int. Conf. Robotics Autom. (ICRA), Scottsdale (1989) pp. 125–130

10.77 S. Chiaverini, B. Siciliano, O. Egeland: Kinematic analysis and singularity avoidance for a seven-joint manipulator, Am. Control Conf., San Diego (1990) pp. 2300–2305

10.78 D.R. Baker, C.W. Wampler: On the inverse kinematics of redundant manipulators, Int. J. Robotics Res. **7**(2), 3–21 (1988)

10.79 J.W. Burdick: On the inverse kinematics of redundant manipulators: Characterization of the self-motion manifolds, IEEE Int. Conf. Robotics Autom. (ICRA), Scottsdale (1989) pp. 264–270

10.80 S. Chiaverini, O. Egeland, R.K. Kanestrøm: Weighted damped least-squares in kinematic control of robotic manipulators, Adv. Robotics **7**, 201–218 (1993)

10.81 M. Kirćanski, M. Vukobratović: Trajectory planning for redundant manipulators in presence of obstacles, 5th CISM-IFToMM Symp. Theor. Pract. Robots Manip., Udine (1984) pp. 43–58

10.82 C.A. Klein: Use of redundancy in the design of robotic systems. In: *Robotics Research – The Second International Symposium*, ed. by H. Hanafusa, H. Hinoue (MIT Press, Cambridge 1985) pp. 207–214

10.83 J. Baillieul: Kinematic redundancy and the control of robots with flexible components, IEEE Int. Conf. Robotics Autom. (ICRA), Nice (1992) pp. 715–721

10.84 I.D. Walker: The use of kinematic redundancy in reducing impact and contact effects in manipulation, IEEE Int. Conf. Robotics Autom. (ICRA), Cincinnati (1990) pp. 434–439

10.85 M.W. Gertz, J.O. Kim, P.K. Khosla: Exploiting redundancy to reduce impact force, IEEE/RSJ Int. Workshop Intell. Robots Syst., Osaka (1991) pp. 179–184

10.86 T. Yoshikawa: Dynamic manipulability of robot manipulators, J. Robotics Syst. **2**(1), 113–124 (1985)

10.87 G. Oriolo, M. Ottavi, M. Vendittelli: Probabilistic motion planning for redundant robots along given end-effector paths, IEEE/RSJ Int. Conf. Intell. Robots Syst., Lausanne (2002) pp. 1657–1662

10.88 D.N. Nenchev: Redundancy resolution through local optimization: A review, J. Robotics Syst. **6**, 6 (1989)

10.89 A. De Luca, G. Oriolo: The reduced gradient method for solving redundancy in robot arms, Robotersysteme **7**(2), 117–122 (1991)

10.90 S. Seereeram, J.T. Wen: A global approach to path planning for redundant manipulators, IEEE Trans. Robotics Autom. **11**(1), 152–160 (1995)

10.91 A. De Luca, G. Oriolo, B. Siciliano: Robot redundancy resolution at the acceleration level, Lab. Robotics Autom. **4**(2), 97–106 (1992)

10.92 O. Khatib: Inertial properties in robotics manipulation: An object-level framework, Int. J. Robotics Res. **14**(1), 19–36 (1995)

10.93 C.A. Klein, C.H. Huang: Review of pseudoinverse control for use with kinematically redundant manipulators, IEEE Trans. Syst. Man Cybern. **13**, 245–250 (1983)

10.94 R. Mukherjee: Design of holonomic loops for repeatability in redundant manipulators, IEEE Int. Conf. Robotics Autom. (ICRA), Nagoya (1985) pp. 2785–2790

10.95 A. De Luca, G. Oriolo: Nonholonomic behavior in redundant robots under kinematic control, IEEE Trans. Robotics Autom. **13**(5), 776–782 (1997)

10.96 B. Bayle, J.Y. Fourquet, M. Renaud: Manipulability of wheeled-mobile manipulators: Application to motion generation, Int. J. Robotics Res. **22**(7/8), 565–581 (2003)

10.97 A. De Luca, G. Oriolo, P.R. Giordano: Kinematic modeling and redundancy resolution for nonholonomic mobile manipulators, IEEE Int. Conf. Robotics Autom. (ICRA), Orlando (2006) pp. 1867–1873

Multimedia Contents

11. Robots with Flexible Elements

Alessandro De Luca, Wayne J. Book

Design issues, dynamic modeling, trajectory planning, and feedback control problems are presented for robot manipulators having components with mechanical flexibility, either concentrated at the joints or distributed along the links. The chapter is divided accordingly into two main parts. Similarities or differences between the two types of flexibility are pointed out wherever appropriate.

For robots with flexible joints, the dynamic model is derived in detail by following a Lagrangian approach and possible simplified versions are discussed. The problem of computing the nominal torques that produce a desired robot motion is then solved. Regulation and trajectory tracking tasks are addressed by means of linear and nonlinear feedback control designs.

For robots with flexible links, relevant factors that lead to the consideration of distributed flexibility are analyzed. Dynamic models are presented, based on the treatment of flexibility through lumped elements, transfer matrices, or assumed modes. Several specific issues are then highlighted, including the selection of sensors, the model order used for control design, and the generation of effective commands that reduce or eliminate residual vibrations in rest-to-rest maneuvers. Feedback control alternatives are finally discussed.

In each of the two parts of this chapter, a section is devoted to the illustration of the original references and to further readings on the subject.

11.1 **Robots with Flexible Joints** 244
11.1.1 Dynamic Modeling 245
11.1.2 Inverse Dynamics 249
11.1.3 Regulation Control 252
11.1.4 Trajectory Tracking 257
11.1.5 Further Reading 261

11.2 **Robots with Flexible Links** 263
11.2.1 Design Issues 263
11.2.2 Modeling of Flexible Link Arms 265
11.2.3 Control 269
11.2.4 Further Reading 278

Video-References 279

References 279

The standard assumption underlying robot kinematics, dynamics, and control design is that manipulators consist only of rigid bodies (links and motion transmission components). This is, however, an ideal situation that may be considered valid only for slow motion and small interacting forces. In practice, mechanical flexibility in robot manipulators is present for two main reasons: the use of compliant transmission elements and the reduction of the mass of the moving links through the use of lightweight material and slender design. These two kinds of flexibility introduce static and dynamic deflections between the position of the driving actuators and the position of the manipulator end-effector. If flexibility is not taken into account when considering robot design and control, a degradation of the overall expected performance of the robot typically occurs.

From a modeling point of view, flexibility can be assumed as concentrated at the robot joints or distributed (in different ways) along the robot links. The dynamic modeling steps are similar, with the need to introduce additional generalized coordinates besides those used to describe the rigid motion of the robot arm. However, the properties of the resulting models are quite different from a control point of view. Therefore, the case of *robots with flexible joints* and of *robots with flexible links* are presented in this chapter mostly as

separate items, pointing out similarities or structural differences wherever appropriate. For both classes of flexible robots, the relevant design issues, dynamic modeling, inverse dynamics algorithms, and control laws for set-point regulation and trajectory tracking will be discussed. Indeed, joint and link flexibilities may be present (and dynamically interact) at the same time. Many of the presented results can be extended to this case as well.

In the following, it is assumed that the reader is already familiar with basic issues on kinematics, dynamics, and control of rigid robots (Chaps. 2, 3, 8).

11.1 Robots with Flexible Joints

The presence of joint flexibility is common in current industrial robots, when motion transmission/reduction elements such as belts – (as in the selective compliant assembly robot arm (SCARA) family), long shafts (as in the forearm of the Unimation Puma arm), cables, harmonic drives, or cycloidal gears are used. The purpose of these components is to allow relocation of the actuators next to the robot base, thus improving dynamic efficiency, or to guarantee high reduction ratios with power-efficient compact inline devices.

However, when subject to the forces/torques arising in normal robot operation, these components are intrinsically flexible (e.g., the flexspline in harmonic drives; Fig. 11.1) and introduce a time-varying displacement between the position of the actuators and that of the driven links. Without a specific control action, an oscillatory behavior, typically of small magnitude but at a relatively high frequency, is observed at the robot end-effector level during free motion. Moreover, some form of instability (e.g., chattering) may occur in tasks involving contact with the environment.

Recently, flexible actuation/transmission elements have been deliberately selected in soft robots (such as those driven by serial elastic actuation (SEA) or variable stiffness actuation (VSA), Chap. 21) intended for physical human–robot interaction and for achieving more natural motion characteristics. In fact, this form of mechanical compliance guarantees an inertial decoupling between the actuator and the (possibly, lightweight) link, thus reducing the kinetic energy involved in undesired collisions with humans. Such a safety-oriented mechanical design should be traded-off with a more complex control design aimed at preserving the same performance of rigid robots in terms of speed and accuracy of the end-effector motion (Chap. 69).

Figures 11.2 and 11.3 show two examples of robot manipulators with flexible joints, the 8*R* robot *Dex-*

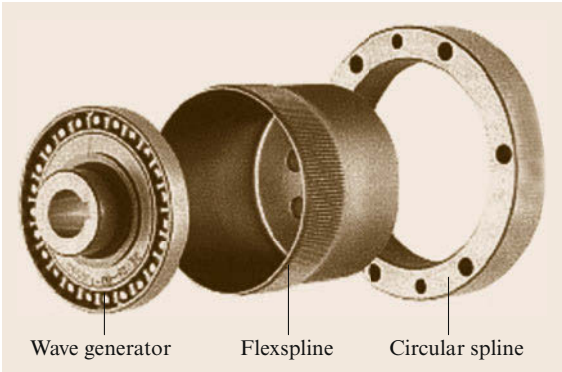


Fig. 11.1 The components of a harmonic drive

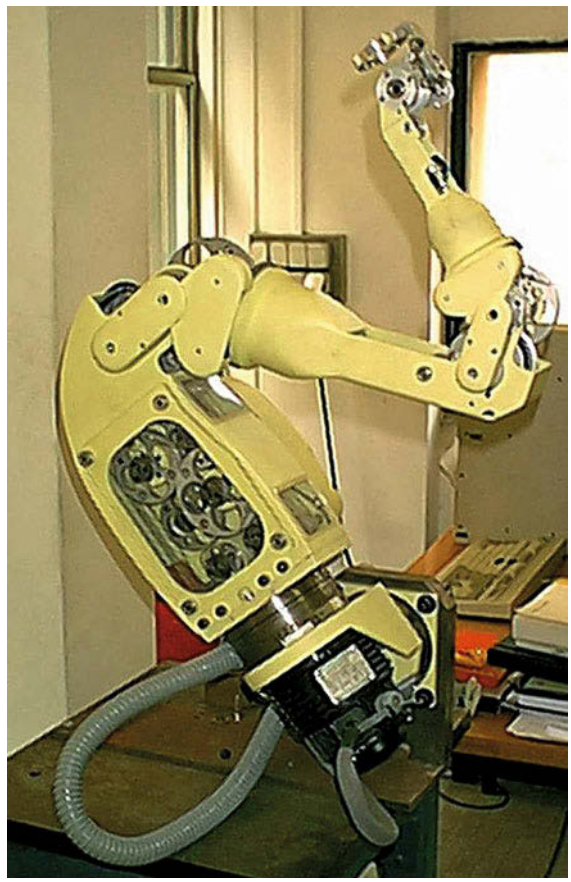


Fig. 11.2 The cable-driven robot *Dexter* by Scienzia Machinale

ter and the 7*R* lightweight manipulator *DLR LWR-III*. The first robot has motors 3–8 located inside the second link, with motion transmitted to the distal links through steel cables and pulleys; the second robot has a modular structure, with each motor integrated in the associated joint and using an harmonic drive as reduction element. The joint stiffnesses of the *Dexter* robot vary between 120 and 6300 N m/rad depending on the joint, while for the *DLR LWR-III* manipulator, the range is 6000–15 000 N m/rad. These numerical values refer to deflections and associated torques evaluated after the reduction gears.

Deflection of the flexible transmission elements can be modeled as being *concentrated at the joints* of the robot, thus limiting the complexity of the associated equations of motion. When compared to the rigid case, the dynamic model of robots with flexible joints (and rigid links) requires *twice* the number of generalized coordinates to completely characterize the configuration of all rigid bodies (motors and links) constituting the arm.

As a result, the case of flexible joint robots is one example in which the number of control inputs is strictly less than the number of mechanical degrees of freedom. This suggests that the design of control schemes realizing standard motion tasks is expected to be more difficult than for rigid robots. Moreover, the implementation of a full state feedback law will require twice the number of sensors, measuring quantities that are *before* and *after* (or across) the joint deformation. On the other hand, the motor torques used to command the robot as well as the disturbance torques due to joint flexibility perform work on the same joint axes (they are *physically collocated*). It should be noted that this is one relevant difference with respect to the case of flexibility distributed along the links. This characteristic is very helpful for rejecting vibrations and controlling the overall robot motion. Roughly speaking, we will be able to use input commands acting before the source of flexibility to ensure that output variables defined beyond the flexibility behave in a desired way.



Fig. 11.3 The lightweight manipulator *DLR LWR-III* by the German Aerospace Center

11.1.1 Dynamic Modeling

Dynamic models that explicitly include joint flexibility are used to evaluate quantitatively vibratory effects superposed on the rigid motion, verify whether the control laws obtained under the assumption of joint rigidity can still work in practice (or to what extent they should be modified), and enable the design of new model-based feedforward and feedback controllers.

We consider a robot with flexible joints as an open kinematic chain having $N+1$ rigid bodies, the base and the N links, interconnected by N (rotary or prismatic) joints undergoing deflection, and actuated by N electrical drives. From a mechanical point of view, each motor (with its stator and rotor) is an additional rigid body with its inertial properties. All joints are considered to be flexible, though mixed situations may be encountered due to the use of different transmission devices. When reduction gearings are present, they are modeled as being placed *before* the joint deflection occurs.

The following standard assumptions are made:

A1 Joint deflections are small, so that flexibility effects are limited to the domain of linear elasticity.
A2 The actuators' rotors are modeled as uniform bodies having their center of mass on the rotation axis.
A3 Each motor is located on the robot arm in a position preceding the driven link. (This can be generalized to the case of multiple motors simultaneously driving multiple distal links.)

The first assumption supports the terminology of robots with *elastic* joints often used in the literature. The elasticity at joint i is modeled by a spring of stiff-

Avidbots Ex1016
Page 289 of 2154

ness $K_i > 0$, which is torsional for rotational joints and linear for translational ones. Figure 11.4 shows a single link driven by a motor through a rotational elastic joint. Nonlinear stiffness characteristics for the flexible joint can also be considered, provided that the map from deflection to force is smooth and invertible. Assumption **A2** is a basic requirement for the long life of an electrical drive, and thus very reasonable. As we will see, it implies that the inertia matrix and the gravity term in the robot dynamic model will be independent of the angular position of the motors. A typical arrangement of motors satisfying assumption **A3** is shown in Fig. 11.5. The simplest situation is when the i-th motor moves link i and is mounted on link $i-1$ with its rotation axis aligned with the i-th joint. This is, for instance, the case for the *LWR-III* manipulator. The dislocation of the actuating motors along the structure has a great influence on the structure of the dynamic equations of motion.

For kinematic and dynamic analysis, $2N$ frames are attached to the $2N$ moving rigid bodies (links and motors) in the robot chain: the link frames L_i and the motor frames R_i, for $i = 1, \ldots, N$ (Fig. 11.5). For the definition of the link frames L_i, the standard Denavit–Hartenberg convention can be used. The frames R_i are attached to the motor stators, aligned with the axes of symmetry of the motors and with the z-axis along the spinning direction of the rotor.

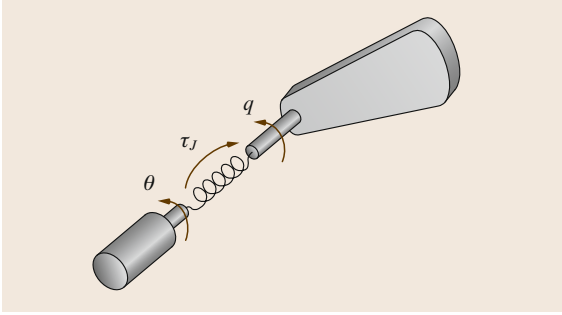


Fig. 11.4 Schematic representation of an elastic joint

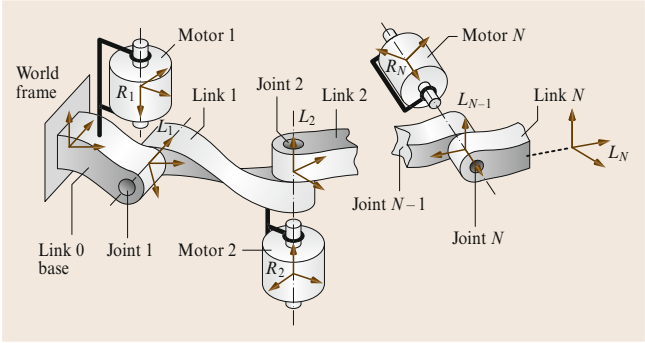


Fig. 11.5 Arrangement of motors and links in an open kinematic chain

Accordingly, $2N$ generalized coordinates will be needed. A possible set of coordinates is given by

$$\boldsymbol{\Theta} = \begin{pmatrix} \boldsymbol{q} \\ \boldsymbol{\theta} \end{pmatrix} \in \mathbb{R}^{2N} ,$$

where $\boldsymbol{q}$ is the N-vector of link positions and $\boldsymbol{\theta}$ is the N-vector of motor (i. e., rotor) positions, as reflected through the transmission/reduction gears. This choice of variables is quite convenient because:

1. The model will be formally independent of the reduction ratios.
2. These position variables will have a similar dynamic range.
3. The kinematics of the robot will be a function of the link variables $\boldsymbol{q}$ only (these variables are already *beyond* the joint flexibility) so that all issues related to direct/inverse kinematics will be identical to the case of fully rigid robots.

For some considerations, it is also useful to define the variable $\boldsymbol{\theta}_{\mathrm{m}}$, namely the N-vector of motor positions before reduction, which are the quantities directly measured by encoders mounted on the motors. In the case of a motor directly placed on the i-th joint axis, one has $\dot{\theta}_{\mathrm{m},i} = n_i \dot{\theta}_i$, where $n_i \geq 1$ is the reduction ratio at the i-th joint. In addition, for $i = 1, \ldots, N$, the difference

$$\delta_i = q_i - \theta_i$$

is the deflection at the i-th joint, while

$$\tau_{\mathrm{J},i} = K_i(\theta_i - q_i)$$

is the torque transmitted to the i-th link through the associated elastic spring (Fig. 11.4) – the quantity measured by a joint torque sensor, whenever present. Note that for robots with flexible links, the set $(\boldsymbol{\theta}, \boldsymbol{\delta})$ is typically used in the dynamic modeling, where $\boldsymbol{\delta}$ is the vector of link deflection coordinates.

Following a Lagrangian approach, the single energy contributions to the Lagrangian

$$\mathcal{L} = \mathcal{T}(\boldsymbol{\Theta}, \dot{\boldsymbol{\Theta}}) - \mathcal{U}(\boldsymbol{\Theta})$$

will be derived next.

The *potential energy* of the robot is due to gravity and joint elasticity. The gravity part is related to the position of the barycenter of the links (each of mass m_i) and of the motors (of mass m_{r_i}). Because of assumption **A2**, the latter will be independent of $\boldsymbol{\theta}$. Thus,

$$\mathcal{U}_{\mathrm{grav}} = \mathcal{U}_{\mathrm{grav,link}}(\boldsymbol{q}) + \mathcal{U}_{\mathrm{grav,motor}}(\boldsymbol{q}) .$$

Avidbots Ex1016
Page 290 of 2154

For the elastic part of the potential, because of assumption **A1**, we have

$$\mathcal{U}_{\text{elas}} = \frac{1}{2}(\boldsymbol{q}-\boldsymbol{\theta})^{\mathrm{T}}\mathbf{K}(\boldsymbol{q}-\boldsymbol{\theta}) ,$$
$$\mathbf{K} = \operatorname{diag}(K_1,\dots,K_N) .$$

As a result,

$$\mathcal{U}(\boldsymbol{\Theta}) = \mathcal{U}_{\text{grav}}(\boldsymbol{q}) + \mathcal{U}_{\text{elas}}(\boldsymbol{q}-\boldsymbol{\theta}) .$$

The *kinetic energy* of the robot is the sum of the link and rotor contributions. For the links, there is no difference with respect to the standard rigid robot case and it will be sufficient to write in general

$$\mathcal{T}_{\text{link}} = \frac{1}{2}\dot{\boldsymbol{q}}^{\mathrm{T}}\mathbf{M}_{\mathrm{L}}(\boldsymbol{q})\dot{\boldsymbol{q}} , \tag{11.1}$$

with the positive-definite symmetric link inertia matrix $\mathbf{M}_{\mathrm{L}}(\boldsymbol{q})$. For the rotors, some more details are needed

$$\begin{aligned}\mathcal{T}_{\text{rotor}} &= \sum_{i=1}^{N}\mathcal{T}_{\text{rotor}_i} \\ &= \sum_{i=1}^{N}\left(\frac{1}{2}m_{r_i}\boldsymbol{v}_{r_i}^{\mathrm{T}}\boldsymbol{v}_{r_i} + \frac{1}{2}{}^{R_i}\boldsymbol{\omega}_{r_i}^{\mathrm{T}}{}^{R_i}\mathbf{I}_{r_i}{}^{R_i}\boldsymbol{\omega}_{r_i}\right),\end{aligned} \tag{11.2}$$

where $\boldsymbol{v}_{r_i}$ is the linear velocity of the center of mass of the i-th rotor and $\boldsymbol{\omega}_{r_i}$ is the angular velocity of the i-th rotor body. All quantities in the angular contributions in (11.2) are conveniently expressed in the local frame R_i. According to assumption **A2**, the rotor inertia matrix is then diagonal

$${}^{R_i}\mathbf{I}_{r_i} = \operatorname{diag}(I_{r_{ixx}}, I_{r_{iyy}}, I_{r_{izz}}) ,$$

with $I_{r_{ixx}} = I_{r_{iyy}}$, while $\boldsymbol{v}_{r_i}$ can be expressed as a function of $\boldsymbol{q}$ and $\dot{\boldsymbol{q}}$ only. Moreover, due to assumption **A3**, the angular velocity of the i-th rotor has the general expression

$${}^{R_i}\boldsymbol{\omega}_{r_i} = \sum_{j=1}^{i-1}\mathbf{J}_{r_i,j}(\boldsymbol{q})\dot{q}_j + \begin{pmatrix}0\\0\\ \dot{\theta}_{m,i}\end{pmatrix}, \tag{11.3}$$

where $\mathbf{J}_{r_i,j}(\boldsymbol{q})$ is the j-th column of the Jacobian relating the link velocities $\dot{\boldsymbol{q}}$ to the angular velocity of the i-th rotor in the robot chain. By substituting (11.3) into (11.2) and expressing $\dot{\boldsymbol{\theta}}_m$ in terms of $\dot{\boldsymbol{\theta}}$, it can be shown that

$$\begin{aligned}\mathcal{T}_{\text{rotor}} = &\frac{1}{2}\dot{\boldsymbol{q}}^{\mathrm{T}}\left[\mathbf{M}_{\mathrm{R}}(\boldsymbol{q}) + \mathbf{S}(\boldsymbol{q})\mathbf{B}^{-1}\mathbf{S}^{\mathrm{T}}(\boldsymbol{q})\right]\dot{\boldsymbol{q}} \\ &+ \dot{\boldsymbol{q}}^{\mathrm{T}}\mathbf{S}(\boldsymbol{q})\dot{\boldsymbol{\theta}} + \frac{1}{2}\dot{\boldsymbol{\theta}}^{\mathrm{T}}\mathbf{B}\,\dot{\boldsymbol{\theta}} ,\end{aligned} \tag{11.4}$$

where $\mathbf{B}$ is the constant diagonal inertia matrix collecting the rotors inertial components $I_{r_{izz}}$ around their spinning axes, as reflected through the reduction ratios, $\mathbf{M}_{\mathrm{R}}(\boldsymbol{q})$ contains the rotor masses (and, possibly, the rotor inertial components along the other principal axes), and the square matrix $\mathbf{S}(\boldsymbol{q})$ expresses the inertial couplings between the rotors and the previous links in the robot chain.

A simple example illustrates the derivation and the actual expressions of the terms in (11.4). We also include the reduction elements to show how the rotor inertial component around the spinning axis will appear in the kinetic energy, weighted by different powers of the reduction ratios. Consider a planar robot with two rotational elastic joints, a first link of length ℓ_1, and motors mounted directly on the joint axes. The kinetic energies of the two rotors are

$$\begin{aligned}\mathcal{T}_{\text{rotor}_1} &= \frac{1}{2}I_{r_{1zz}}\dot{\theta}_{m,1}^2 = \frac{1}{2}I_{r_{1zz}}n_1^2\dot{\theta}_1^2 \\ \mathcal{T}_{\text{rotor}_2} &= \frac{1}{2}m_{r_2}\ell_1^2\dot{q}_1^2 + \frac{1}{2}I_{r_{2zz}}(\dot{q}_1+\dot{\theta}_{m,2})^2 \\ &= \frac{1}{2}m_{r_2}\ell_1^2\dot{q}_1^2 + \frac{1}{2}I_{r_{2zz}}(\dot{q}_1^2 + 2n_2\dot{q}_1\dot{\theta}_2 + n_2^2\dot{\theta}_2^2) ,\end{aligned}$$

leading to

$$\mathbf{B} = \begin{pmatrix}I_{r_{1zz}}n_1^2 & 0\\ 0 & I_{r_{2zz}}n_2^2\end{pmatrix}, \qquad \mathbf{S} = \begin{pmatrix}0 & I_{r_{2zz}}n_2\\ 0 & 0\end{pmatrix},$$
$$\mathbf{M}_{\mathrm{R}} = \begin{pmatrix}m_{r_2}\ell_1^2 & 0\\ 0 & 0\end{pmatrix}, \qquad \mathbf{S}\mathbf{B}^{-1}\mathbf{S}^{\mathrm{T}} = \begin{pmatrix}I_{r_{2zz}} & 0\\ 0 & 0\end{pmatrix}.$$

In this specific case, the matrix $\mathbf{S}$ (as well as $\mathbf{M}_{\mathrm{R}}$) is constant. Note that for large reduction ratios n_i, the dominant inertial effect due to the rotors is given by the matrix $\mathbf{B}$. Also, if the second motor was mounted remotely on the first joint (as it is often the case in the class of SCARA arms), or still close to the second joint but with the spinning axis orthogonal to the axis of this joint, then the matrix $\mathbf{S}$ would be zero.

In general, as a consequence of assumption **A3**, the matrix $\mathbf{S}(\boldsymbol{q})$ has always a strictly upper triangular structure with a cascaded dependence of its nonzero elements

$$\mathbf{S}(\boldsymbol{q}) = \begin{pmatrix}
0 & S_{12} & S_{13}(q_2) & S_{14}(q_2,q_3) & \dots & \dots & S_{1N}(q_2,\dots,q_{N-1})\\
0 & 0 & S_{23} & S_{24}(q_3) & \dots & \dots & S_{2N}(q_3,\dots,q_{N-1})\\
0 & 0 & 0 & S_{34} & \dots & \dots & S_{3N}(q_4,\dots,q_{N-1})\\
\vdots & \vdots & \vdots & \ddots & \ddots & \vdots & \vdots\\
0 & 0 & 0 & \dots & 0 & S_{N-2,N-1} & S_{N-2,N}(q_{N-1})\\
0 & 0 & 0 & \dots & 0 & 0 & S_{N-1,N}\\
0 & 0 & 0 & \dots & 0 & 0 & 0
\end{pmatrix}. \tag{11.5}$$

Summing up, the total kinetic energy of the robot is

$$\mathcal{T} = \frac{1}{2}\dot{\boldsymbol{\Theta}}^{\mathrm{T}}\boldsymbol{\mathcal{M}}(\boldsymbol{\Theta})\dot{\boldsymbol{\Theta}} = \frac{1}{2}\begin{pmatrix}\dot{\boldsymbol{q}}^{\mathrm{T}} & \dot{\boldsymbol{\theta}}^{\mathrm{T}}\end{pmatrix}\begin{pmatrix}\mathbf{M}(\boldsymbol{q}) & \mathbf{S}(\boldsymbol{q})\\ \mathbf{S}^{\mathrm{T}}(\boldsymbol{q}) & \mathbf{B}\end{pmatrix}\begin{pmatrix}\dot{\boldsymbol{q}}\\ \dot{\boldsymbol{\theta}}\end{pmatrix},$$

with

$$\mathbf{M}(\boldsymbol{q}) = \mathbf{M}_{\mathrm{L}}(\boldsymbol{q}) + \mathbf{M}_{\mathrm{R}}(\boldsymbol{q}) + \mathbf{S}(\boldsymbol{q})\mathbf{B}^{-1}\mathbf{S}^{\mathrm{T}}(\boldsymbol{q})\,. \quad (11.6)$$

As anticipated, the total inertia matrix $\boldsymbol{\mathcal{M}}$ of the robot depends only on $\boldsymbol{q}$.

Using the Lagrange equations, finally yields the complete dynamic model

$$\begin{pmatrix}\mathbf{M}(\boldsymbol{q}) & \mathbf{S}(\boldsymbol{q})\\ \mathbf{S}^{\mathrm{T}}(\boldsymbol{q}) & \mathbf{B}\end{pmatrix}\begin{pmatrix}\ddot{\boldsymbol{q}}\\ \ddot{\boldsymbol{\theta}}\end{pmatrix} + \begin{pmatrix}\boldsymbol{c}(\boldsymbol{q},\dot{\boldsymbol{q}}) + \boldsymbol{c}_1(\boldsymbol{q},\dot{\boldsymbol{q}},\dot{\boldsymbol{\theta}})\\ \boldsymbol{c}_2(\boldsymbol{q},\dot{\boldsymbol{q}})\end{pmatrix} + \begin{pmatrix}\boldsymbol{g}(\boldsymbol{q}) + \mathbf{K}(\boldsymbol{q}-\boldsymbol{\theta})\\ \mathbf{K}(\boldsymbol{\theta}-\boldsymbol{q})\end{pmatrix} = \begin{pmatrix}\mathbf{0}\\ \boldsymbol{\tau}\end{pmatrix}, \quad (11.7)$$

where the inertial terms (related to the total inertia matrix $\boldsymbol{\mathcal{M}}(\boldsymbol{q})$), the Coriolis and centrifugal terms (collectively denoted by $\boldsymbol{c}_{\mathrm{tot}}(\boldsymbol{\Theta},\dot{\boldsymbol{\Theta}})$), and the potential terms $(\partial\mathcal{U}(\boldsymbol{\Theta})/\partial\boldsymbol{\Theta})^{\mathrm{T}}$ have been written separately. In particular, $\boldsymbol{g}(\boldsymbol{q}) = (\partial\mathcal{U}_{\mathrm{grav}}(\boldsymbol{q})/\partial\boldsymbol{q})^{\mathrm{T}}$ while $\boldsymbol{\tau}_{\mathrm{J}} = \mathbf{K}(\boldsymbol{\theta}-\boldsymbol{q})$ is the elastic torque transmitted through the joints.

The first N and the last N equations of the dynamic model (11.7) are referred to as the *link* and the *motor equations*, respectively.

On the right-hand side of (11.7), all nonconservative generalized forces should appear. When dissipative effects are not considered, only the motor torque $\boldsymbol{\tau}$ producing work on the $\boldsymbol{\theta}$ variable is present (i. e., the torques on the motor output shafts, amplified by the reduction ratios) in the motor equations. If the robot end-effector is in contact with the environment, the zero on the right-hand side of the link equations should be replaced by $\boldsymbol{\tau}_{\mathrm{ext}} = \mathbf{J}^{\mathrm{T}}(\boldsymbol{q})\mathbf{F}$, where $\mathbf{J}(\boldsymbol{q})$ is the robot Jacobian and $\mathbf{F}$ the forces/torques acting from the environment on the robot.

In the presence of energy-dissipating effects, additional terms appear on the right-hand side of (11.7). For instance, viscous friction at both sides of the transmissions and spring damping of the (visco)elastic joints give rise to the vector term

$$\begin{pmatrix}-\mathbf{F}_{\boldsymbol{q}}\dot{\boldsymbol{q}} - \mathbf{D}(\dot{\boldsymbol{q}}-\dot{\boldsymbol{\theta}})\\ -\mathbf{F}_{\boldsymbol{\theta}}\dot{\boldsymbol{\theta}} - \mathbf{D}(\dot{\boldsymbol{\theta}}-\dot{\boldsymbol{q}})\end{pmatrix}, \quad (11.8)$$

where the diagonal, positive-definite matrices $\mathbf{F}_{\boldsymbol{q}}$, $\mathbf{F}_{\boldsymbol{\theta}}$, and $\mathbf{D}$ contain, respectively, the viscous coefficients on the link side and on the motor side, and the damping of the elastic springs at the joints. More general forms of nonlinear friction $\boldsymbol{\tau}_{\mathrm{F}}$ can be considered. Note that, in principle, friction acting on the motor side can be fully compensated by a suitable choice of the control torque $\boldsymbol{\tau}$, while this is not true for friction acting on the link side, due to the noncollocation.

Model Properties

All elements in the $2N$-vector $\boldsymbol{c}_{\mathrm{tot}}(\boldsymbol{\Theta},\dot{\boldsymbol{\Theta}})$ of velocity-dependent terms in (11.7) are independent of the motor position $\boldsymbol{\theta}$. The specific dependence of the N-vectors $\boldsymbol{c}$, $\boldsymbol{c}_1$, and $\boldsymbol{c}_2$ follows from the general expression of the components of $\boldsymbol{c}_{\mathrm{tot}}$ based on Christoffel symbols

$$c_{\mathrm{tot},i}(\boldsymbol{\Theta},\dot{\boldsymbol{\Theta}}) = \frac{1}{2}\dot{\boldsymbol{\Theta}}^{\mathrm{T}}\left[\frac{\partial\boldsymbol{\mathcal{M}}_i}{\partial\boldsymbol{\Theta}} + \left(\frac{\partial\boldsymbol{\mathcal{M}}_i}{\partial\boldsymbol{\Theta}}\right)^{\mathrm{T}} - \frac{\partial\boldsymbol{\mathcal{M}}}{\partial\boldsymbol{\Theta}_i}\right]\dot{\boldsymbol{\Theta}}\,,$$

for $i = 1,\ldots,2N$, where $\boldsymbol{\mathcal{M}}_i$ is the i-th column of the total inertia matrix $\boldsymbol{\mathcal{M}}(\boldsymbol{\Theta})$. In particular, the velocity vectors $\boldsymbol{c}_1$ and $\boldsymbol{c}_2$ arise only in the presence of a configuration-dependent $\mathbf{S}(\boldsymbol{q})$ matrix. Performing calculations, it can be shown that:

1. The vector $\boldsymbol{c}_1$ does not contain quadratic velocity terms in $\dot{\boldsymbol{q}}$ or $\dot{\boldsymbol{\theta}}$, but only mixed quadratic terms $\dot{\theta}_i\dot{q}_j$.
2. When the matrix $\mathbf{S}$ is constant (in particular, zero), both $\boldsymbol{c}_1$ and $\boldsymbol{c}_2$ vanish.

The dynamic model (11.7) also shares some properties of the rigid case, such as:

- The model equations admit a linear parametrization in terms of a suitable set of dynamic coefficients, including joint stiffnesses and motor inertias, which is useful for model identification and adaptive control.
- The Coriolis and centrifugal terms can always be factorized as $\boldsymbol{c}_{\mathrm{tot}}(\boldsymbol{\Theta},\dot{\boldsymbol{\Theta}}) = \boldsymbol{C}(\boldsymbol{\Theta},\dot{\boldsymbol{\Theta}})\dot{\boldsymbol{\Theta}}$, in such a way that matrix $\dot{\boldsymbol{\mathcal{M}}} - 2\boldsymbol{C}$ is skew-symmetric – a property used in control analysis.
- For robots having only rotational joints, the gradient of the gravity vector $\boldsymbol{g}(\boldsymbol{q})$ is globally bounded in norm by a constant.

Finally, when the joint stiffness is extremely large ($\mathbf{K}\to\infty$), then $\boldsymbol{\theta}\to\boldsymbol{q}$ while $\boldsymbol{\tau}_{\mathrm{J}}\to\boldsymbol{\tau}$. It is easy to check that the dynamic model (11.7) collapses in the limit into the standard model of fully rigid robots (including links and motors).

Reduced Model

In general, the link and motor equations in (11.7) are dynamically coupled through the elastic torque $\boldsymbol{\tau}_{\mathrm{J}}$ at the joints, but also (at the acceleration level) via the inertial components of matrix $\mathbf{S}(\boldsymbol{q})$ – usually a path of low energy transfer. The presence and actual relevance

of these inertial couplings depend on the kinematic arrangement of the manipulator and, in particular, on the specific placement of the motors and transmission devices. There are cases in which the matrix $\mathbf{S}$ is constant (e.g., the planar case, as in the previous example, with any number of links) or zero (e.g., for a single link with elastic joint or for a robot with $N = 2$ links having the two joint axes orthogonal and the motors mounted at the joints). As a result, the dynamic equations will simplify considerably.

For a generic robot with elastic joints, one can take advantage of the presence of large reduction ratios (with the n_i's of the order of 100–150) and simply neglect energy contributions due to the inertial couplings between the motors and the links (see again the $2R$ planar example). This is equivalent to considering the following simplifying assumption:

A4 The angular velocity of the rotors is due only to their own spinning, i. e.,

$$ {}^{R_i}\boldsymbol{\omega}_{r_i} = \begin{pmatrix} 0 & 0 & \dot{\theta}_{m,i} \end{pmatrix}^{\mathrm{T}} , \quad i = 1, \ldots, N , $$

instead of the complete expression (11.3).

As a result, the total angular kinetic energy of the rotors is just $\frac{1}{2}\dot{\boldsymbol{\theta}}^{\mathrm{T}}\mathbf{B}\dot{\boldsymbol{\theta}}$ (or $\mathbf{S} \equiv \mathbf{0}$), and the dynamic model (11.7) reduces to

$$ \begin{aligned} \mathbf{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{c}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \boldsymbol{g}(\boldsymbol{q}) + \mathbf{K}(\boldsymbol{q} - \boldsymbol{\theta}) &= \mathbf{0} \\ \mathbf{B}\ddot{\boldsymbol{\theta}} + \mathbf{K}(\boldsymbol{\theta} - \boldsymbol{q}) &= \boldsymbol{\tau} , \end{aligned} \qquad (11.9) $$

with $\mathbf{M}(\boldsymbol{q}) = \mathbf{M}_{\mathrm{L}}(\boldsymbol{q}) + \mathbf{M}_{\mathrm{R}}(\boldsymbol{q})$. The main feature of this model is that the link and motor equations are dynamically coupled only through the elastic torque $\boldsymbol{\tau}_J$. In addition, the motor equations are now fully linear.

We note that the complete model (11.7) and the reduced model (11.9) display different characteristics with respect to certain control problems. In fact, the reduced model is always feedback linearizable by *static* state feedback, whereas this is never the case for the complete model as soon as a coupling $\mathbf{S} \neq \mathbf{0}$ is present.

Singular Perturbation Model

It is interesting to show the *two-time-scale* dynamic behavior that it is induced in robots with elastic joints when the joint stiffness $\mathbf{K}$ is relatively large, but still finite. This behavior can be made explicit by a simple linear change of coordinates, namely replacing $\boldsymbol{\theta}$ with the joint torque $\boldsymbol{\tau}_{\mathrm{J}}$. For the sake of simplicity, this is illustrated on the reduced model only (without dissipation).

Since the diagonal joint stiffness matrix is assumed to have large but similar elements, a common scalar factor $1/\epsilon^2 \gg 1$ can be extracted as

$$ \mathbf{K} = \frac{1}{\epsilon^2}\hat{\mathbf{K}} = \frac{1}{\epsilon^2}\,\mathrm{diag}\left(\hat{K}_1, \ldots, \hat{K}_N\right) . $$

The *slow* subsystem is then given by the link equations, once they are rewritten as

$$ \mathbf{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{c}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \boldsymbol{g}(\boldsymbol{q}) = \boldsymbol{\tau}_{\mathrm{J}} . \qquad (11.10) $$

To obtain the dynamics of the *fast* subsystem, the joint torque is differentiated twice, the motor and link accelerations are replaced from (11.9), and the above definition of $\hat{\mathbf{K}}$ is used. This leads to

$$ \begin{aligned} \epsilon^2\ddot{\boldsymbol{\tau}}_{\mathrm{J}} = \hat{\mathbf{K}} \{ & \mathbf{B}^{-1}\boldsymbol{\tau} - \left[\mathbf{B}^{-1} + \mathbf{M}^{-1}(\boldsymbol{q})\right]\boldsymbol{\tau}_{\mathrm{J}} \\ & + \mathbf{M}^{-1}(\boldsymbol{q})[\boldsymbol{c}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \boldsymbol{g}(\boldsymbol{q})] \} . \end{aligned} \qquad (11.11) $$

For small ϵ, (11.10) and (11.11) represent a singularly perturbed system. The two separate time scales governing the slow and fast dynamics are t and $\sigma = t/\epsilon$, since

$$ \epsilon^2\,\ddot{\boldsymbol{\tau}}_{\mathrm{J}} = \epsilon^2\,\frac{\mathrm{d}^2\boldsymbol{\tau}_{\mathrm{J}}}{\mathrm{d}t^2} = \frac{\mathrm{d}^2\boldsymbol{\tau}_{\mathrm{J}}}{\mathrm{d}\sigma^2} . $$

This model serves as a basis for composite control schemes, where the control torque has the general form

$$ \boldsymbol{\tau} = \boldsymbol{\tau}_{\mathrm{s}}(\boldsymbol{q}, \dot{\boldsymbol{q}}, t) + \epsilon\boldsymbol{\tau}_{\mathrm{f}}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{\tau}_{\mathrm{J}}, \dot{\boldsymbol{\tau}}_{\mathrm{J}}) . \qquad (11.12) $$

This includes a slow action $\boldsymbol{\tau}_{\mathrm{s}}$, designed when neglecting joint elasticity, and an additional action $\boldsymbol{\tau}_{\mathrm{f}}$ for locally stabilizing the fast flexible dynamics around a suitable manifold in the state space. It can be verified that, when setting $\epsilon = 0$ in (11.10)–(11.12), the equivalent rigid robot model is recovered as

$$ [\mathbf{M}(\boldsymbol{q}) + \mathbf{B}]\,\ddot{\boldsymbol{q}} + \boldsymbol{c}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \boldsymbol{g}(\boldsymbol{q}) = \boldsymbol{\tau}_{\mathrm{s}} . $$

A similar singular perturbation model (and control design) can also be derived for robot manipulators with flexible links.

11.1.2 Inverse Dynamics

Given a desired motion of a robot, we wish to compute the nominal torque needed to reproduce this motion in ideal conditions (the inverse dynamics problem). Such nominal torque can be used as the feedforward term in a trajectory-tracking control law.

For rigid robots, the inverse dynamics is a straightforward algebraic computation obtained by replacing the desired motion of the generalized coordinates in the dynamic model. The minimum requirement for exact

reproduction is that the planned motion has a continuously differentiable desired velocity. For robots with elastic joints, a motion task can be conveniently expressed in terms of a desired link trajectory $\boldsymbol{q} = \boldsymbol{q}_{\mathrm{d}}(t)$ (possibly obtained from the kinematic inversion of a desired motion in Cartesian space). The additional complexity lies in the fact that not all robot coordinates are directly assigned in this way, so that additional derivations should be performed. This will require some higher degree of smoothness of the desired trajectory $\boldsymbol{q}_{\mathrm{d}}(t) \in [0, T]$, where the final time T may or may not be finite.

Reduced Model

Consider first the case of the reduced model (11.7) and set $\boldsymbol{n}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \boldsymbol{c}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \boldsymbol{g}(\boldsymbol{q})$ for compactness. By evaluating the link equations on the desired link motion

$$\mathbf{M}(\boldsymbol{q}_{\mathrm{d}})\ddot{\boldsymbol{q}}_{\mathrm{d}} + \boldsymbol{n}\left(\boldsymbol{q}_{\mathrm{d}}, \dot{\boldsymbol{q}}_{\mathrm{d}}\right) + \mathbf{K}\boldsymbol{q}_{\mathrm{d}} = \mathbf{K}\boldsymbol{\theta}_{\mathrm{d}}\,, \tag{11.13}$$

the nominal position $\boldsymbol{\theta}_{\mathrm{d}}$ of the motors associated with the desired link motion is readily obtained. The nominal elastic torque at the joints is $\boldsymbol{\tau}_{\mathrm{J,d}} = \mathbf{K}(\boldsymbol{\theta}_{\mathrm{d}} - \boldsymbol{q}_{\mathrm{d}})$; note that, from (11.13), this quantity can be expressed as a function of $\boldsymbol{q}_{\mathrm{d}}$, $\dot{\boldsymbol{q}}_{\mathrm{d}}$, and $\ddot{\boldsymbol{q}}_{\mathrm{d}}$, which is independent of $\mathbf{K}$. Differentiating (11.13) leads to the expression for the nominal motor velocity $\dot{\boldsymbol{\theta}}_{\mathrm{d}}$,

$$\mathbf{M}(\boldsymbol{q}_{\mathrm{d}})\boldsymbol{q}_{\mathrm{d}}^{[3]} + \dot{\mathbf{M}}(\boldsymbol{q}_{\mathrm{d}})\ddot{\boldsymbol{q}}_{\mathrm{d}} + \dot{\boldsymbol{n}}\left(\boldsymbol{q}_{\mathrm{d}}, \dot{\boldsymbol{q}}_{\mathrm{d}}\right) + \mathbf{K}\dot{\boldsymbol{q}}_{\mathrm{d}} = \mathbf{K}\dot{\boldsymbol{\theta}}_{\mathrm{d}}\,, \tag{11.14}$$

where the notation $\boldsymbol{y}^{[i]} = \mathrm{d}^i\boldsymbol{y}/\mathrm{d}t^i$ is used. Differentiating once more, we obtain

$$\begin{aligned}\mathbf{M}(\boldsymbol{q}_{\mathrm{d}})\boldsymbol{q}_{\mathrm{d}}^{[4]} + 2\dot{\mathbf{M}}(\boldsymbol{q}_{\mathrm{d}})\boldsymbol{q}_{\mathrm{d}}^{[3]} + \ddot{\boldsymbol{n}}\left(\boldsymbol{q}_{\mathrm{d}}, \dot{\boldsymbol{q}}_{\mathrm{d}}\right) \\ + \left[\ddot{\mathbf{M}}(\boldsymbol{q}_{\mathrm{d}}) + \mathbf{K}\right]\ddot{\boldsymbol{q}}_{\mathrm{d}} = \mathbf{K}\ddot{\boldsymbol{\theta}}_{\mathrm{d}}\,,\end{aligned} \tag{11.15}$$

to be used in the motor equations evaluated along the desired motion. After simplifications, the nominal torque is obtained as

$$\begin{aligned}\boldsymbol{\tau}_{\mathrm{d}} = {}& [\mathbf{M}(\boldsymbol{q}_{\mathrm{d}}) + \mathbf{B}]\,\ddot{\boldsymbol{q}}_{\mathrm{d}} + \boldsymbol{n}\left(\boldsymbol{q}_{\mathrm{d}}, \dot{\boldsymbol{q}}_{\mathrm{d}}\right) \\ & + \mathbf{B}\mathbf{K}^{-1}\left[\mathbf{M}(\boldsymbol{q}_{\mathrm{d}})\boldsymbol{q}_{\mathrm{d}}^{[4]} + 2\dot{\mathbf{M}}(\boldsymbol{q}_{\mathrm{d}})\boldsymbol{q}_{\mathrm{d}}^{[3]}\right. \\ & \left. + \ddot{\mathbf{M}}(\boldsymbol{q}_{\mathrm{d}})\ddot{\boldsymbol{q}}_{\mathrm{d}} + \ddot{\boldsymbol{n}}\left(\boldsymbol{q}_{\mathrm{d}}, \dot{\boldsymbol{q}}_{\mathrm{d}}\right)\right],\end{aligned} \tag{11.16}$$

where the extra contributions to the nominal torque of the rigid case due to joint elasticity can be clearly recognized. The evaluation of $\boldsymbol{\tau}_{\mathrm{d}}$ involves the computation of first- and second-order partial derivatives of the dynamic model terms. For instance, one needs to compute

$$\dot{\mathbf{M}}[\boldsymbol{q}_{\mathrm{d}}(t)] = \sum_{i=1}^{N} \left.\frac{\partial \mathbf{M}_i(\boldsymbol{q})}{\partial \boldsymbol{q}}\right|_{\boldsymbol{q}=\boldsymbol{q}_{\mathrm{d}}(t)} \dot{\boldsymbol{q}}_{\mathrm{d}}(t)\boldsymbol{e}_i^{\mathrm{T}}\,,$$

where $\boldsymbol{e}_i$ is the i-th unit vector and $\boldsymbol{M}_i$ is the i-th column of matrix $\mathbf{M}(\boldsymbol{q})$. This and other similar expressions can be obtained by symbolic manipulation software. From (11.16), it follows that the minimum requirement for the exact reproducibility of the desired motion is that $\boldsymbol{q}_{\mathrm{d}}(t)$ admits a continuously differentiable jerk (i. e., that $\boldsymbol{q}_{\mathrm{d}}^{[4]}(t)$ exists in the time interval $[0, T]$). Such a smoother requirement should not come unexpected, in view of the flexible nature of the system.

Similarly to the rigid case, a recursive numerical Newton–Euler algorithm can be defined, which computes efficiently the inverse dynamics torque (11.16) in $O(N)$ asymptotic complexity. For the reduced model of robots with elastic joints, the algorithm involves forward recursion of motion variables up to the fourth differential order and backward recursion of second time derivatives of forces and torques.

Complete Model

Some more analysis is needed for the model (11.9). For ease of exposition, and without loss of generality, the case of a constant matrix $\mathbf{S}$ is considered. When evaluating the link equations on the desired link motion

$$\mathbf{M}(\boldsymbol{q}_{\mathrm{d}})\ddot{\boldsymbol{q}}_{\mathrm{d}} + \mathbf{S}\ddot{\boldsymbol{\theta}}_{\mathrm{d}} + \boldsymbol{n}\left(\boldsymbol{q}_{\mathrm{d}}, \dot{\boldsymbol{q}}_{\mathrm{d}}\right) + \mathbf{K}\boldsymbol{q}_{\mathrm{d}} = \mathbf{K}\boldsymbol{\theta}_{\mathrm{d}}\,, \tag{11.17}$$

the additional presence of the motor acceleration on the left-hand side does not allow solving directly for the motor position $\boldsymbol{\theta}_{\mathrm{d}}$ as a function of $(\boldsymbol{q}_{\mathrm{d}}, \dot{\boldsymbol{q}}_{\mathrm{d}}, \ddot{\boldsymbol{q}}_{\mathrm{d}})$ only. However, the strictly upper triangular structure (11.5) of $\mathbf{S}$ allows one to define $\boldsymbol{\theta}_{\mathrm{d}}$ componentwise, using the scalar equations in (11.17) recursively. The N-th equation is in fact independent of $\ddot{\boldsymbol{\theta}}_{\mathrm{d}}$,

$$\mathbf{M}_N^{\mathrm{T}}(\boldsymbol{q}_{\mathrm{d}})\ddot{\boldsymbol{q}}_{\mathrm{d}} + \mathbf{0}^{\mathrm{T}}\ddot{\boldsymbol{\theta}}_{\mathrm{d}} + n_N\left(\boldsymbol{q}_{\mathrm{d}}, \dot{\boldsymbol{q}}_{\mathrm{d}}\right) + K_N q_{\mathrm{d},N} = K_N\theta_{\mathrm{d},N}\,,$$

so that this equation can be used to define

$$\theta_{\mathrm{d},N} = f_N\left(\boldsymbol{q}_{\mathrm{d}}, \dot{\boldsymbol{q}}_{\mathrm{d}}, \ddot{\boldsymbol{q}}_{\mathrm{d}}\right)$$

and, after double differentiation, its second time derivative

$$\ddot{\theta}_{\mathrm{d},N} = f_N''\left(\boldsymbol{q}_{\mathrm{d}}, \dot{\boldsymbol{q}}_{\mathrm{d}}, \ldots, \boldsymbol{q}_{\mathrm{d}}^{[4]}\right).$$

In the $(N-1)$-th equation,

$$\begin{aligned}\mathbf{M}_{N-1}^{\mathrm{T}}(\boldsymbol{q}_{\mathrm{d}})\ddot{\boldsymbol{q}}_{\mathrm{d}} + S_{N-1,N}\ddot{\theta}_{\mathrm{d},N} \\ + n_{N-1}\left(\boldsymbol{q}_{\mathrm{d}}, \dot{\boldsymbol{q}}_{\mathrm{d}}\right) + K_{N-1}q_{\mathrm{d},N-1} = K_{N-1}\theta_{\mathrm{d},N-1}\,,\end{aligned}$$

the acceleration $\ddot{\theta}_{\mathrm{d},N}$ has already been determined in the previous step. Therefore, this equation can be used similarly to define

$$\theta_{\mathrm{d},N-1} = f_{N-1}\left(\boldsymbol{q}_{\mathrm{d}}, \dot{\boldsymbol{q}}_{\mathrm{d}}, \ldots, \boldsymbol{q}_{\mathrm{d}}^{[4]}\right)$$

and, after two time differentiations, also

$$\ddot{\theta}_{\mathrm{d},N-1} = f''_{N-1}\left(\boldsymbol{q}_\mathrm{d}, \dot{\boldsymbol{q}}_\mathrm{d}, \ldots, \boldsymbol{q}_\mathrm{d}^{[6]}\right).$$

Note that whenever $S_{N-1,N} = 0$, there will be no increase in the degree of the highest order derivatives of $\boldsymbol{q}_\mathrm{d}$ within the functional dependence of $\theta_{\mathrm{d},N-1}$. This argument also applies recursively to the following steps. Proceeding backward through the link equations, the scalar computations end up with the definition of

$$\theta_{\mathrm{d},1} = f_1\left(\boldsymbol{q}_\mathrm{d}, \dot{\boldsymbol{q}}_\mathrm{d}, \ldots, \boldsymbol{q}_\mathrm{d}^{[2N]}\right)$$

and

$$\ddot{\theta}_{\mathrm{d},1} = f''_1\left(\boldsymbol{q}_\mathrm{d}, \dot{\boldsymbol{q}}_\mathrm{d}, \ldots, \boldsymbol{q}_\mathrm{d}^{[2(N+1)]}\right),$$

where the dependence on the highest possible differential degree of $\boldsymbol{q}_\mathrm{d}$ is shown. With $\ddot{\boldsymbol{\theta}}_\mathrm{d} = \boldsymbol{f}''(\cdot)$ made available in this way, the nominal torque is finally computed from the motor equations, again evaluated along the desired motion. Using (11.17) to substitute for $\mathbf{K}(\boldsymbol{\theta}_\mathrm{d} - \boldsymbol{q}_\mathrm{d})$ yields

$$\begin{aligned}\boldsymbol{\tau}_\mathrm{d} = &\left[\mathbf{M}(\boldsymbol{q}_\mathrm{d}) + \mathbf{S}^\mathrm{T}\right]\ddot{\boldsymbol{q}}_\mathrm{d} + \boldsymbol{n}\left(\boldsymbol{q}_\mathrm{d}, \dot{\boldsymbol{q}}_\mathrm{d}\right) \\ &+ (\mathbf{B} + \mathbf{S})\ \ddot{\boldsymbol{\theta}}_\mathrm{d}\left(\boldsymbol{q}_\mathrm{d}, \dot{\boldsymbol{q}}_\mathrm{d}, \ldots, \boldsymbol{q}_\mathrm{d}^{[2(N+1)]}\right).\end{aligned} \tag{11.18}$$

As a result, the presence of inertial motor–link couplings considerably increases the complexity of the solution to the inverse dynamics problem. The exact reproduction of a desired link trajectory by the nominal torque in (11.18) requires that $\boldsymbol{q}_\mathrm{d}(t)$ has a continuously differentiable $(2N+1)$-th time derivative (i. e., that $\boldsymbol{q}_\mathrm{d}^{[2(N+1)]}(t)$ exists in the time interval $[0, T]$). For rest-to-rest motions, this implies a trajectory plan that imposes a very slow start and ending phases.

We finally remark that the possibility of expressing the evolution of the state and input of a system algebraically in terms of the evolution of an output variable (in our case $\boldsymbol{q}$) and a finite number of its derivatives is sometimes referred to as the *flatness* property. The above inverse dynamics analysis shows that $\boldsymbol{q}$ is a flat output for robots with elastic joints modeled either by (11.7) or (11.9).

Note also that, when a *constant* $\boldsymbol{q}_\mathrm{d}$ is considered, these computations all provide the same condition for the associated motor position

$$\boldsymbol{\theta}_\mathrm{d} = \boldsymbol{q}_\mathrm{d} + \mathbf{K}^{-1}\boldsymbol{g}(\boldsymbol{q}_\mathrm{d}) \tag{11.19}$$

and nominal static torque

$$\boldsymbol{\tau}_\mathrm{d} = \boldsymbol{g}(\boldsymbol{q}_\mathrm{d}). \tag{11.20}$$

Presence of Dissipative Terms

Inclusion of dissipative terms in the inverse dynamics deserves some additional comments. Any model of frictional effects acting on the motor side of the transmissions can be included in the computation without extra requirements. Friction at the link side should instead be described by a smooth model, because of the need to differentiate the link equations. Thus, some functional approximations may be needed (e.g., replacing discontinuous *sign* functions with hyperbolic tangents) before including this term in (11.13) or (11.17).

On the other hand, the presence of a nonnegligible spring damping $\mathbf{D}$ in (11.8) changes the structure of the computations. While this reduces the order of the derivatives of $\boldsymbol{q}_\mathrm{d}$ involved, the problem itself will become nonalgebraic; in fact, the inverse system will require the use of the solution of a dynamical system, albeit a simple one.

Consider for instance the model (11.9), including now all the dissipative terms given in (11.8). When evaluating the link equations,

$$\begin{aligned}\mathbf{M}(\boldsymbol{q}_\mathrm{d})\ddot{\boldsymbol{q}}_\mathrm{d} + \boldsymbol{n}\left(\boldsymbol{q}_\mathrm{d}, \dot{\boldsymbol{q}}_\mathrm{d}\right) + \left(\mathbf{D} + \mathbf{F}_q\right)\dot{\boldsymbol{q}}_\mathrm{d} + \mathbf{K}\boldsymbol{q}_\mathrm{d} \\ = \mathbf{D}\dot{\boldsymbol{\theta}}_\mathrm{d} + \mathbf{K}\boldsymbol{\theta}_\mathrm{d},\end{aligned} \tag{11.21}$$

the motor velocity $\dot{\boldsymbol{\theta}}_\mathrm{d}$ additionally appears on the right-hand side. Differentiating (11.21) leads to

$$\mathbf{D}\ddot{\boldsymbol{\theta}}_\mathrm{d} + \mathbf{K}\dot{\boldsymbol{\theta}}_\mathrm{d} = \boldsymbol{w}_\mathrm{d}, \tag{11.22}$$

with

$$\begin{aligned}\boldsymbol{w}_\mathrm{d} = &\mathbf{M}(\boldsymbol{q}_\mathrm{d})\boldsymbol{q}_\mathrm{d}^{[3]} + \left[\dot{\mathbf{M}}(\boldsymbol{q}_\mathrm{d}) + \mathbf{D} + \mathbf{F}_\mathrm{q}\right]\ddot{\boldsymbol{q}}_\mathrm{d} \\ &+ \dot{\boldsymbol{n}}\left(\boldsymbol{q}_\mathrm{d}, \dot{\boldsymbol{q}}_\mathrm{d}\right) + \mathbf{K}\dot{\boldsymbol{q}}_\mathrm{d}.\end{aligned}$$

Equation (11.22) represents a first-order linear *asymptotically stable* dynamical system (internal dynamics), with state $\dot{\boldsymbol{\theta}}_\mathrm{d}$ and forcing signal $\boldsymbol{w}_\mathrm{d}(t)$. For a given $\dot{\boldsymbol{\theta}}_\mathrm{d}(0)$, its solution $\dot{\boldsymbol{\theta}}_\mathrm{d}(t)$ and the associated solution derivative $\ddot{\boldsymbol{\theta}}_\mathrm{d}(t)$ are needed to evaluate the nominal torque in the motor equations. This yields

$$\boldsymbol{\tau}_\mathrm{d} = \mathbf{M}(\boldsymbol{q}_\mathrm{d})\ddot{\boldsymbol{q}}_\mathrm{d} + \boldsymbol{n}\left(\boldsymbol{q}_\mathrm{d}, \dot{\boldsymbol{q}}_\mathrm{d}\right) + \mathbf{F}_\mathrm{q}\dot{\boldsymbol{q}}_\mathrm{d} + \mathbf{B}\ddot{\boldsymbol{\theta}}_\mathrm{d} + \mathbf{F}_{\boldsymbol{\theta}}\dot{\boldsymbol{\theta}}_\mathrm{d},$$

where (11.21) has been used to replace the term $\mathbf{D}(\dot{\boldsymbol{\theta}}_\mathrm{d} - \dot{\boldsymbol{q}}_\mathrm{d}) + \mathbf{K}(\boldsymbol{\theta}_\mathrm{d} - \boldsymbol{q}_\mathrm{d})$. In this case, the desired link trajectory $\boldsymbol{q}_\mathrm{d}(t)$ should have a continuously differentiable acceleration ($\boldsymbol{q}_\mathrm{d}^{[3]}$ should exist, since it is used in the definition of $\boldsymbol{w}_\mathrm{d}$). Note that any initialization $\dot{\boldsymbol{\theta}}_\mathrm{d}(0)$ of the internal dynamics (11.22) is feasible (i. e., it produces a specific torque profile $\boldsymbol{\tau}_\mathrm{d}(t)$ yielding the same link

Avidbots Ex1016
Page 295 of 2154

motion $\boldsymbol{q}_{\mathrm{d}}(t)$), with the associated motor position $\boldsymbol{\theta}_{\mathrm{d}}(0)$ initialized from (11.21). Indeed, we should match the actual initial state of the robot. For instance, starting from an equilibrium state implies the unique choice $\dot{\boldsymbol{\theta}}_{\mathrm{d}}(0) = \mathbf{0}$.

A similar procedure can also be applied to the complete model (11.7) in the presence of spring damping. Again a dynamical inverse system will be needed, while the smoothness requirement on $\boldsymbol{q}_{\mathrm{d}}(t)$ would be even more dramatically reduced in that case.

We finally remark that, for robots with flexible links, the inverse dynamics problem gives also rise to an internal dynamics, independently of the presence of modal damping. Moreover, when specifying a desired motion for the tip of the flexible arm, the associated internal dynamics will be *unstable* and this critical issue has to be tackled to determine a feasible solution.

11.1.3 Regulation Control

We now consider the problem of controlling the motion of a robot with joint elasticity to a *constant* configuration. In this problem, no trajectory planning is involved and a feedback law should be found that achieves asymptotic stabilization of a desired closed-loop equilibrium. Global solutions (i. e., those valid when starting from any initial state) are indeed preferred.

From the analysis in the previous section, it should be clear that one needs to define only a constant reference $\boldsymbol{q}_{\mathrm{d}}$ (with $\dot{\boldsymbol{q}}_{\mathrm{d}}(t) \equiv \mathbf{0}$) for the link coordinates.

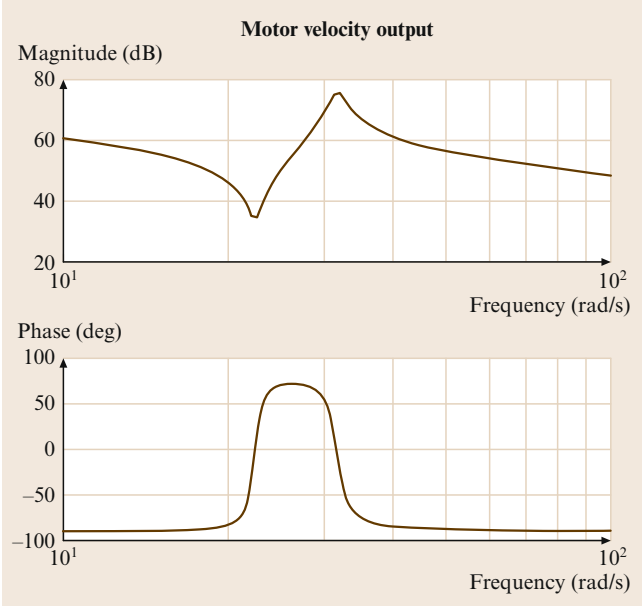


Fig. 11.6 Bode diagrams of the torque to motor velocity transfer function

From (11.19), a unique reference $\boldsymbol{\theta}_{\mathrm{d}}$ for the motor variables is in fact associated with the desired $\boldsymbol{q}_{\mathrm{d}}$ (which, in turn, may result from a desired pose of the robot end-effector). Furthermore, (11.20) provides the needed static torque to be applied at steady state by any feasible controller.

A major aspect of the presence of joint elasticity is that the feedback part of the control law can depend in general on *four* variables for each joint: the motor and link position, and the motor and link velocity. However, in most robots where joint elasticity is not explicitly considered in the system design, at most two sensors are available for measurement at the joints: a position sensor (e.g., an encoder) and, in some cases, a tachometer as a velocity sensor. When no velocity sensors are present, velocity is typically reconstructed by suitable numerical differentiation of high-resolution position measurements. Due to the presence of joint elasticity, the position/velocity quantities that are actually measured depend on where these sensors are mounted in the motor/transmission assembly.

A single link driven through an elastic joint is introduced as a paradigmatic case study to show what can be achieved using different sets of partial state measurements. This simple situation provides indications on how to handle the general multilink case.

In the absence of gravity, it will be shown that a proportional–derivative (PD) controller based only on motor measurements is sufficient to achieve the desired regulation task. In the presence of gravity, various gravity compensation schemes can be added to the PD feedback controller. These control results mimic the situation of robots with rigid joints, once the convenient quantities for use in the feedback have been identified.

Single Elastic Joint Example

Consider a single link rotating on a horizontal plane (thus, without gravity) and actuated with a motor through an elastic joint coupling (Fig. 11.4). When viscous friction on motor and link side as well as damping across the spring are included, the dynamic model is

$$M\ddot{q} + D(\dot{q} - \dot{\theta}) + K(q - \theta) + F_q\dot{q} = 0\,,$$
$$B\ddot{\theta} + D(\dot{\theta} - \dot{q}) + K(\theta - q) + F_\theta\dot{\theta} = \tau\,,$$

where the same, now scalar, notation of Sect. 11.1.1 has been used. Since this system is described by linear equations, Laplace transforms are used to derive the transfer functions of interest, namely from input torque to motor position

$$\frac{\theta(s)}{\tau(s)} = \frac{Ms^2 + (D + F_q)s + K}{\mathrm{den}(s)}\,,$$

Avidbots Ex1016
Page 296 of 2154

and from input torque to link position

$$\frac{q(s)}{\tau(s)} = \frac{Ds + K}{\mathrm{den}(s)} ,$$

with the common denominator den(s) given by

$$\begin{aligned}\mathrm{den}(s) = \Big\{ & MB\, s^3 + \big[M(D + F_\theta) + B(D + F_q)\big]\, s^2 \\ & + \big[(M + B)K + (F_q + F_\theta)D + F_q F_\theta\big]\, s \\ & + (F_q + F_\theta)K \Big\} s .\end{aligned}$$

In the case of link position output, the transfer function has a larger relative degree (or pole-zero excess). Figures 11.6 and 11.7 show typical frequency responses in the two cases. For clarity, the outputs have been chosen at the velocity level. In the Bode diagram of the magnitude for the motor velocity output, note the presence of an antiresonance/resonance behavior. Similarly, there is a pure resonance (more pronounced for weak or zero spring damping D) for the link velocity output. The phase profile in Fig. 11.7, with a high-frequency lag of 270°, indicates the greater control difficulty encountered when closing the feedback loop on link quantities. In experimental tests on a robot joint, such plots are quite characteristic of the presence of concentrated joint elasticity and can be used to assess the relevance of this phenomenon and for identifying model parameters.

For the analysis of the stabilizing properties of different feedback arrangements, all dissipative effects will be neglected in the following ($D = F_q = F_\theta = 0$ is anyway the worst case). Consider first the transfer function relative to the motor position output

$$\left.\frac{\theta(s)}{\tau(s)}\right|_{\text{no diss}} = \frac{Ms^2 + K}{[MB\, s^2 + (M + B)K]\, s^2} . \qquad (11.23)$$

Apart from a double pole at the origin, this transfer function has a pair of imaginary zeros and poles, with the zero pair occurring at the so-called locked frequency

$$\omega_1 = \sqrt{\frac{K}{M}} ,$$

which characterizes the oscillations occurring when the motor is locked ($\theta \equiv 0$), e.g., by a high-gain positional feedback. This frequency is used to assess the performance limit of a simple PD control for the motor variables. In order to achieve enough damping in the closed-loop system, the bandwidth should be limited, as a rule of thumb, to one-third of ω_1. Faster transients can be achieved only by taking into account the fourth-order dynamics of the elastic joint assembly. Note also

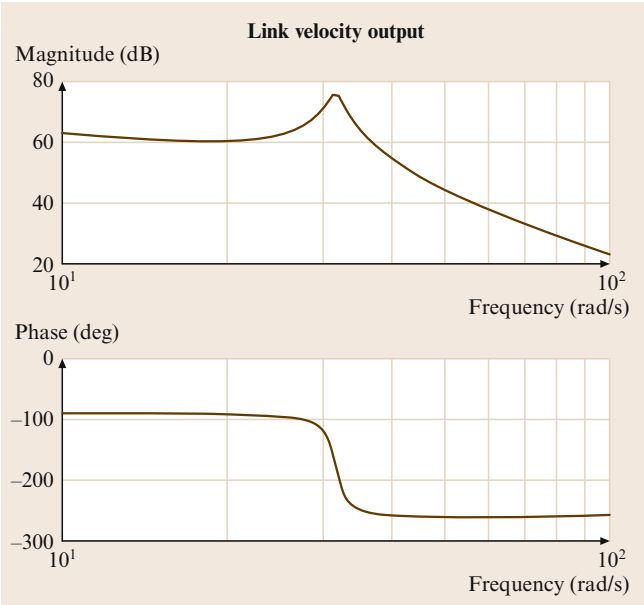


Fig. 11.7 Bode diagrams of the torque to link velocity transfer function

that the frequency of the zero pair is always *lower* than that of the pole pair in (11.23). This is related to the *passivity* of the mapping from torque to motor velocity, a property useful for stability and robust or adaptive control design.

Since the control objective is to regulate the link position output, we are also interested in the open-loop transfer function

$$\left.\frac{q(s)}{\tau(s)}\right|_{\text{no diss}} = \frac{K}{[MB\, s^2 + (M + B)K]\, s^2} . \qquad (11.24)$$

This transfer function has *no zeros* (in fact, for this to happen it is sufficient that $D = 0$) so that the relative degree is now four, the maximum possible. We shall see that the nonlinear counterpart of the absence of zeros in (11.24) will play a relevant role also for trajectory control in the multilink case of robots with elastic joints.

It is worth mentioning that this situation is completely different from the case in which the elastic spring would be moved out of the joint and placed anywhere along the link – a simple one-mode approximate model of link flexibility. In that case, the analogous transfer function will possess a negative and a positive real zero, symmetrically placed w.r.t. the origin (a *nonminimum-phase* system). This indicates the criticality of a direct inversion of the system input–output map for the execution of a desired link trajectory.

With the desired position given in terms of the link variable q_d, the most natural choice for the design of

Avidbots Ex1016
Page 297 of 2154

a linear stabilizing feedback using one position and one velocity variable, is to close a PD loop from the link variables,

$$\tau = u_q - \left(K_{P,q} q + K_{D,q} \dot{q}\right) , \tag{11.25}$$

with position and velocity gains $K_{P,q}$ and $K_{D,q}$, and with $u_q = K_{P,q} q_\mathrm{d}$ being the external input used for defining the set point. It is easy to verify that the closed-loop poles will be unstable no matter how the gains are chosen, so that error feedback from link variables alone should be avoided. In a similar way, also the combination of motor position and link velocity feedback is always unstable.

Another mixed feedback strategy is to use link position and motor velocity

$$\tau = u_q - \left(K_{P,q} q + K_{D,\mathrm{m}} \dot{\theta}\right) . \tag{11.26}$$

This combination corresponds, e.g., to the case of a tachometer integrated in a direct-current (DC) motor and of an optical encoder placed on the load shaft for sensing its position (without any knowledge about the relevance of joint elasticity). Use of (11.26) leads to the closed-loop characteristic equation

$$\begin{aligned} &BMs^4 + MK_{D,\mathrm{m}} s^3 + (B+M)Ks^2 \\ &\quad + KK_{D,\mathrm{m}} s + KK_{P,q} = 0 . \end{aligned}$$

Using Routh's criterion, asymptotic stability occurs if and only if the motor velocity gain $K_{D,\mathrm{m}} > 0$ and the link position gain satisfies $0 < K_{P,q} < K$, i. e., the proportional feedback should not *override* the spring stiffness. The existence of such an upper bound limits the usefulness of this scheme.

Finally, consider performing feedback from the motor variables

$$\tau = u_\theta - \left(K_{P,\mathrm{m}} \theta + K_{D,\mathrm{m}} \dot{\theta}\right) , \tag{11.27}$$

with $u_\theta = K_{P,\mathrm{m}} \theta_d = K_{P,\mathrm{m}} q_d$ (due to the absence of gravity). The closed-loop system will be asymptotically stable provided that both $K_{P,\mathrm{m}}$ and $K_{D,\mathrm{m}}$ are strictly positive (and otherwise arbitrarily large). This favorable situation lends itself to a convenient generalization in the multilink case.

Note that other partial state feedback combinations would be possible, depending on the available sensing devices. For instance, mounting a strain gauge on the transmission shaft provides a direct measure of the elastic torque $\tau_\mathrm{J} = K(\theta - q)$ for control use. Strain gauges are also useful sensors for flexible links. Indeed, full-state feedback may be designed so as to guarantee asymptotic stability and improve the transient behavior considerably; however, this would be obtained at the cost of additional sensors and requires proper tuning of the four gains.

PD Control Using only Motor Variables

For the general multilink case in the absence of gravity, consider the PD control law based on motor position and velocity feedback

$$\boldsymbol{\tau} = \mathbf{K}_P(\boldsymbol{\theta}_\mathrm{d} - \boldsymbol{\theta}) - \mathbf{K}_D \dot{\boldsymbol{\theta}} , \tag{11.28}$$

with symmetric (typically, diagonal) positive-definite gain matrices $\mathbf{K}_P$ and $\mathbf{K}_D$. Since $\boldsymbol{g}(\boldsymbol{q}) \equiv \mathbf{0}$, it follows from (11.19) that the reference value for the motor position is $\boldsymbol{\theta}_\mathrm{d} = \boldsymbol{q}_\mathrm{d}$ (neither a joint deflection is present nor an input torque is needed at steady state).

The control law (11.28) globally asymptotically stabilizes the desired equilibrium state $\boldsymbol{q} = \boldsymbol{\theta} = \boldsymbol{q}_\mathrm{d}$, $\dot{\boldsymbol{q}} = \dot{\boldsymbol{\theta}} = \mathbf{0}$. This can be shown through a Lyapunov argument, completed by the application of La Salle's theorem. In fact, a candidate Lyapunov function is given by the sum of the total energy of the system (kinetic plus elastic potential) and of the control energy due to the proportional term (a virtual elastic potential energy)

$$\begin{aligned} V = &\frac{1}{2} \dot{\boldsymbol{\Theta}}^\mathrm{T} \boldsymbol{\mathcal{M}}(\boldsymbol{\Theta}) \dot{\boldsymbol{\Theta}} + \frac{1}{2} (\boldsymbol{q} - \boldsymbol{\theta})^\mathrm{T} \mathbf{K} (\boldsymbol{q} - \boldsymbol{\theta}) \\ &+ \frac{1}{2} (\boldsymbol{\theta}_\mathrm{d} - \boldsymbol{\theta})^\mathrm{T} \mathbf{K}_P (\boldsymbol{\theta}_\mathrm{d} - \boldsymbol{\theta}) \geq 0 . \end{aligned} \tag{11.29}$$

Computing the time derivative of V along the trajectories of the closed-loop system given by (11.7) (or (11.9)) and (11.28), and taking into account the skew-symmetry of $\dot{\boldsymbol{\mathcal{M}}} - 2\boldsymbol{C}$, leads to

$$\dot{V} = -\dot{\boldsymbol{\theta}}^\mathrm{T} \mathbf{K}_D \dot{\boldsymbol{\theta}} \leq 0 .$$

The inclusion of dissipative terms (viscous friction and spring damping) would render $\dot{V}$ even more negative-semidefinite. The analysis is completed by verifying that the maximum invariant set contained in the set of states such that $\dot{V} = 0$ (i. e., those with $\dot{\boldsymbol{\theta}} = \mathbf{0}$) collapses into the desired unique equilibrium state.

We point out that an identical control law is able to globally regulate, in the absence of gravity, robots with flexible links to a desired joint configuration. In that case, $\boldsymbol{\theta}$ in (11.28) would be the rigid coordinates at the base of the flexible links of the robot.

PD with Constant Gravity Compensation

The presence of gravity requires the addition of some form of gravity compensation to the PD action (11.28).

Moreover, this will typically need an additional structural assumption and may demand some caution in the selection of the control gains.

Before proceeding, we recall a basic property of the gravity vector $\boldsymbol{g}(\boldsymbol{q})$ (under assumption **A2**, the gravity vector in (11.7) is the same appearing in the dynamics of the equivalent rigid robot). For robots with rotational joints, elastic or not, a positive constant α exists such that

$$\left\| \frac{\partial \boldsymbol{g}(\boldsymbol{q})}{\partial \boldsymbol{q}} \right\| \leq \alpha \,, \qquad \forall \boldsymbol{q} \in \mathbb{R}^N \,. \tag{11.30}$$

The norm of a matrix $\mathbf{A}(\boldsymbol{q})$ is that induced by the Euclidean norm for vectors, i. e.,

$$\|\mathbf{A}\| = \sqrt{\lambda_{\max}(\mathbf{A}^{\mathrm{T}}\mathbf{A})} \,.$$

Inequality (11.30) implies that

$$\|\boldsymbol{g}(\boldsymbol{q}_1) - \boldsymbol{g}(\boldsymbol{q}_2)\| \leq \alpha \|\boldsymbol{q}_1 - \boldsymbol{q}_2\| \,, \qquad \forall \boldsymbol{q}_1, \boldsymbol{q}_2 \in \mathbb{R}^N \,. \tag{11.31}$$

In common practice, robot joints are never unrealistically soft. More precisely, they are stiff enough to have, under the load of the robot's own weight, a *unique* equilibrium link position $\boldsymbol{q}_{\mathrm{e}}$ associated with any assigned motor position $\boldsymbol{\theta}_{\mathrm{e}}$ – the reverse of the relationship expressed by (11.19). This situation is by no means restrictive, but will be stated for clarity as a further modeling assumption:

A5 The lowest joint stiffness is larger than the upper bound on the gradient of gravity forces acting on the robot, or

$$\min_{i=1,\dots,N} K_i > \alpha \,.$$

The simplest modification for handling the presence of gravity is to consider the addition of a constant term that compensates exactly the gravity load at the desired steady state. According to (11.19) and (11.20), the control law (11.28) is then modified to

$$\boldsymbol{\tau} = \mathbf{K}_P(\boldsymbol{\theta}_{\mathrm{d}} - \boldsymbol{\theta}) - \mathbf{K}_D\dot{\boldsymbol{\theta}} + \boldsymbol{g}(\boldsymbol{q}_{\mathrm{d}}) \,, \tag{11.32}$$

with symmetric, typically diagonal, $\mathbf{K}_P > \mathbf{0}$ (as a minimum) and $\mathbf{K}_D > \mathbf{0}$, and with the motor reference given by $\boldsymbol{\theta}_{\mathrm{d}} = \boldsymbol{q}_{\mathrm{d}} + \mathbf{K}^{-1}\boldsymbol{g}(\boldsymbol{q}_{\mathrm{d}})$.

A sufficient condition guaranteeing that $\boldsymbol{q} = \boldsymbol{q}_{\mathrm{d}}$, $\boldsymbol{\theta} = \boldsymbol{\theta}_{\mathrm{d}}$, $\dot{\boldsymbol{q}} = \dot{\boldsymbol{\theta}} = \mathbf{0}$ will be the unique globally asymptotically stable equilibrium for the system (11.7) under the control law (11.32) is that

$$\lambda_{\min}\left[\begin{pmatrix} \mathbf{K} & -\mathbf{K} \\ -\mathbf{K} & \mathbf{K}+\mathbf{K}_P \end{pmatrix}\right] > \alpha \,, \tag{11.33}$$

with α defined in (11.30). Taking into account the diagonal structure of $\mathbf{K}$ and $\mathbf{K}_P$, and thanks to assumption **A5**, it is always possible to fulfill this condition by increasing the smallest proportional gain in the controller (or, the smallest eigenvalue of $\mathbf{K}_P$ if this matrix has not been chosen diagonal).

In the following, we sketch the motivation for this condition and the associated proof of asymptotic stability. The equilibrium configurations of the closed-loop system are the solutions of

$$\begin{aligned} \mathbf{K}(\boldsymbol{q} - \boldsymbol{\theta}) + \boldsymbol{g}(\boldsymbol{q}) &= \mathbf{0} \,, \\ \mathbf{K}(\boldsymbol{\theta} - \boldsymbol{q}) - \mathbf{K}_P(\boldsymbol{\theta}_{\mathrm{d}} - \boldsymbol{\theta}) - \boldsymbol{g}(\boldsymbol{q}_{\mathrm{d}}) &= \mathbf{0} \,. \end{aligned}$$

Indeed, the pair $(\boldsymbol{q}_{\mathrm{d}}, \boldsymbol{\theta}_{\mathrm{d}})$ satisfies these equations. However, in order to obtain a global result one should guarantee that this pair is the unique solution. Therefore, recalling (11.19), the null term $\mathbf{K}(\boldsymbol{\theta}_{\mathrm{d}} - \boldsymbol{q}_{\mathrm{d}}) - \boldsymbol{g}(\boldsymbol{q}_{\mathrm{d}})$ can be added/subtracted to both equations to obtain

$$\begin{aligned} \mathbf{K}(\boldsymbol{q} - \boldsymbol{q}_{\mathrm{d}}) - \mathbf{K}(\boldsymbol{\theta} - \boldsymbol{\theta}_{\mathrm{d}}) &= \boldsymbol{g}(\boldsymbol{q}_{\mathrm{d}}) - \boldsymbol{g}(\boldsymbol{q}) \\ -\mathbf{K}(\boldsymbol{q} - \boldsymbol{q}_{\mathrm{d}}) + (\mathbf{K} + \mathbf{K}_P)(\boldsymbol{\theta} - \boldsymbol{\theta}_{\mathrm{d}}) &= \mathbf{0} \,, \end{aligned}$$

where the matrix in condition (11.33) can be clearly recognized. Taking the norms of terms on both sides and bounding gravity terms using (11.31), the introduced condition (11.33) implies that the pair $(\boldsymbol{q}_{\mathrm{d}}, \boldsymbol{\theta}_{\mathrm{d}})$ is in fact the unique equilibrium. To show asymptotic stability, a candidate Lyapunov function is built based on the one introduced in (11.29) in the absence of gravity

$$\begin{aligned} V_{\mathrm{g1}} = V + \mathcal{U}_{\mathrm{grav}}(\boldsymbol{q}) - \mathcal{U}_{\mathrm{grav}}(\boldsymbol{q}_{\mathrm{d}}) - (\boldsymbol{q} - \boldsymbol{q}_{\mathrm{d}})^{\mathrm{T}}\boldsymbol{g}(\boldsymbol{q}_{\mathrm{d}}) \\ - \frac{1}{2}\boldsymbol{g}^{\mathrm{T}}(\boldsymbol{q}_{\mathrm{d}})\mathbf{K}^{-1}\boldsymbol{g}(\boldsymbol{q}_{\mathrm{d}}) \geq 0 \,. \end{aligned} \tag{11.34}$$

The positive-definiteness of V_{g1} and the fact that its unique minimum is at the desired state are again implied by condition (11.33). (The last constant term in (11.34) is used to set to zero the minimum value of V_{g1} at the equilibrium.). By the usual computations, it follows that $\dot{V}_{\mathrm{g1}} = -\dot{\boldsymbol{\theta}}^{\mathrm{T}}\mathbf{K}_D\dot{\boldsymbol{\theta}} \leq 0$ and the result is obtained applying La Salle's theorem.

The control law (11.32) is based only on the knowledge of the gravity term $\boldsymbol{g}(\boldsymbol{q}_{\mathrm{d}})$ and of the joint stiffness $\mathbf{K}$. The latter appears in fact in the definition of the motor position reference $\boldsymbol{\theta}_{\mathrm{d}}$. Uncertainties in the gravitational term $\boldsymbol{g}(\boldsymbol{q}_{\mathrm{d}})$ and in the joint stiffness $\mathbf{K}$ affect the performance of the controller. Still, the existence of a unique closed-loop equilibrium and its asymptotic stability are preserved when the gravity bound by α is still correct and condition (11.33) holds for the true stiffness

Avidbots Ex1016
Page 299 of 2154

values. Indeed, the robot will converge to an equilibrium that is different from the desired one – the better the estimates $\hat{\mathbf{K}}$ and $\hat{\boldsymbol{g}}(\boldsymbol{q}_\mathrm{d})$ used, the closer the actual equilibrium will be to the desired one.

PD with Online Gravity Compensation

Similarly to the rigid robot case, a better transient behavior is expected if gravity compensation (or, more precisely, its perfect cancellation) is performed at all configurations during motion. However, the gravity vector in (11.7) depends on the link variables $\boldsymbol{q}$, which are assumed not to be measurable at this stage. It is easy to see that using $\boldsymbol{g}(\boldsymbol{\theta})$, with the measured motor positions in place of the link positions, leads in general to an incorrect closed-loop equilibrium. Moreover, even if $\boldsymbol{q}$ were available, adding $\boldsymbol{g}(\boldsymbol{q})$ to a motor PD error feedback has no guarantee of success because this compensation, which appears in the motor equation, does not instantaneously cancel the gravity load acting on the links.

With this in mind, a PD control with *online* gravity compensation can be introduced as follows. Define the variable

$$\tilde{\boldsymbol{\theta}} = \boldsymbol{\theta} - \mathbf{K}^{-1}\boldsymbol{g}(\boldsymbol{q}_\mathrm{d}) \,, \tag{11.35}$$

which is a *gravity-biased* modification of the measured motor position $\boldsymbol{\theta}$, and let

$$\boldsymbol{\tau} = \mathbf{K}_P\,(\boldsymbol{\theta}_\mathrm{d} - \boldsymbol{\theta}) - \mathbf{K}_D\dot{\boldsymbol{\theta}} + \boldsymbol{g}(\tilde{\boldsymbol{\theta}}) \,, \tag{11.36}$$

where $\mathbf{K}_P > \mathbf{0}$ and $\mathbf{K}_D > \mathbf{0}$ are both symmetric (and typically diagonal) matrices. The control law (11.36) can still be implemented using only motor variables. The term $\boldsymbol{g}(\tilde{\boldsymbol{\theta}})$ provides only an approximate cancellation (though, of a large part) of gravity during motion, but leads to the correct gravity compensation at steady state. In fact, by using (11.19) and (11.35), it follows that

$$\tilde{\boldsymbol{\theta}}_\mathrm{d} := \boldsymbol{\theta}_\mathrm{d} - \mathbf{K}^{-1}\boldsymbol{g}(\boldsymbol{q}_\mathrm{d}) = \boldsymbol{q}_\mathrm{d} \,,$$

so that $\boldsymbol{g}(\tilde{\boldsymbol{\theta}}_\mathrm{d}) = \boldsymbol{g}(\boldsymbol{q}_\mathrm{d})$.

Global asymptotic stability of the desired equilibrium can be guaranteed under the same condition (11.33) used for the constant gravity compensation case. A slightly different Lyapunov candidate is defined, starting again from the one in (11.29), as

$$V_{\mathrm{g}2} = V + \mathcal{U}_\mathrm{grav}(\boldsymbol{q}) - \mathcal{U}_\mathrm{grav}(\tilde{\boldsymbol{\theta}}) - \frac{1}{2}\boldsymbol{g}^\mathrm{T}(\boldsymbol{q}_\mathrm{d})\mathbf{K}^{-1}\boldsymbol{g}(\boldsymbol{q}_\mathrm{d}) \geq 0 \,,$$

to be compared with (11.34).

The use of an online gravity compensation scheme typically provides a smoother time course and a noticeable reduction of the positional transient errors, with no additional control effort in terms of peak and average torques. We note that choices of low position gains, even when largely violating the sufficient condition (11.33) for stability, may still work, contrary to the case of constant gravity compensation. However even for increasing values of joint stiffness, in the limit for $\mathbf{K} \to \infty$, the range of feasible values of $\mathbf{K}_P$ that guarantee exact regulation does not extend down to zero.

A possibility to refine the previous online gravity compensation scheme, again based only on motor position measurement, is offered by the use of a fast iterative algorithm that elaborates the measure $\boldsymbol{\theta}$ in order to generate a quasistatic estimate $\bar{\boldsymbol{q}}(\boldsymbol{\theta})$ of the current (unmeasured) $\boldsymbol{q}$. In fact, in any steady-state configuration $(\boldsymbol{q}_\mathrm{s}, \boldsymbol{\theta}_\mathrm{s})$, a direct mapping is defined from $\boldsymbol{q}_\mathrm{s}$ to $\boldsymbol{\theta}_\mathrm{s}$

$$\boldsymbol{\theta}_\mathrm{s} = \boldsymbol{h}_\mathrm{g}(\boldsymbol{q}_\mathrm{s}) := \boldsymbol{q}_\mathrm{s} + \mathbf{K}^{-1}\boldsymbol{g}(\boldsymbol{q}_\mathrm{s}) \,.$$

Assumption **A5** is sufficient to guarantee the existence and uniqueness also of the inverse mapping $\boldsymbol{q}_\mathrm{s} = \boldsymbol{h}_\mathrm{g}^{-1}(\boldsymbol{\theta}_\mathrm{s})$. For a measured $\boldsymbol{\theta}$, the function

$$\boldsymbol{q} = \boldsymbol{T}(\boldsymbol{q}) := \boldsymbol{\theta} - \mathbf{K}^{-1}\boldsymbol{g}(\boldsymbol{q})$$

is then a *contraction mapping* and the iteration

$$\boldsymbol{q}_{i+1} = \boldsymbol{T}(\boldsymbol{q}_i) \,, \quad i = 0, 1, 2, \ldots$$

will converge to the unique fixed point of this mapping, which is exactly the value $\bar{\boldsymbol{q}}(\boldsymbol{\theta}) = \boldsymbol{h}_\mathrm{g}^{-1}(\boldsymbol{\theta})$. A suitable initialization for $\boldsymbol{q}_0$ is either the measured $\boldsymbol{\theta}$ or the value $\bar{\boldsymbol{q}}$ computed at the previous sampling instant. In this way, just two or three iterations are needed to obtain sufficient accuracy and this is fast enough to be implemented within a sensing/control sampling interval of a digital robot controller. With this iterative scheme running in the background, the regulation control law becomes

$$\boldsymbol{\tau} = \mathbf{K}_P\,(\boldsymbol{\theta}_\mathrm{d} - \boldsymbol{\theta}) - \mathbf{K}_D\dot{\boldsymbol{\theta}} + \boldsymbol{g}\,(\bar{\boldsymbol{q}}(\boldsymbol{\theta})) \,, \tag{11.37}$$

with symmetric (diagonal) $\mathbf{K}_P > \mathbf{0}$ and $\mathbf{K}_D > \mathbf{0}$. A proof of the global asymptotic stability of this control scheme can be given by further modifying the previous Lyapunov candidates. The advantage of (11.37) is that any positive value of the feedback gain $\mathbf{K}_P$ is allowed, thus recovering in full the operative working conditions of the rigid case with exact gravity cancellation.

With reference to the reduced model (11.9), a further improvement can be obtained via a nonlinear PD-like controller that removes completely the gravity from

the actual motion of the robot links. For this, consider the control law

$$\boldsymbol{\tau} = \mathbf{K}_P[\mathbf{q}_d - \boldsymbol{\theta} + \mathbf{K}^{-1}\boldsymbol{g}(\boldsymbol{q})] - \mathbf{K}_D\left(\dot{\boldsymbol{\theta}} - \mathbf{K}^{-1}\dot{\boldsymbol{g}}(\boldsymbol{q})\right) + \boldsymbol{g}(\boldsymbol{q}) + \mathbf{B}\mathbf{K}^{-1}\ddot{\boldsymbol{g}}(\boldsymbol{q}) \tag{11.38}$$

with symmetric (diagonal) $\mathbf{K}_P > 0$ and $\mathbf{K}_D > 0$. The last two terms are those needed to cancel both static and dynamic effects due to gravity. Even when assumption **A5** does not hold, the control law (11.38) guarantees global asymptotic stability of the desired equilibrium state. This result is a byproduct of feedback linearization design (Sect.11.1.4), though much simpler than the complete one, and can be proven using again a Lyapunov analysis. Since a lower bound on the joint stiffness is not required (the joints may be arbitrarily soft), the same approach can be used also for robots using variable stiffness actuation (VSA). The price to pay is that an estimate of the link acceleration $\ddot{\boldsymbol{q}}$ is needed when computing the term $\ddot{\boldsymbol{g}}$ in (11.38).

The two online gravity compensation schemes (11.36) and (11.37) realize a *compliance control* in the joint space with only motor measurements. The same idea can also be extended to the case of Cartesian compliance control by evaluating the direct kinematics and the Jacobian (transpose) of the robot arm with $\tilde{\boldsymbol{\theta}}$ or, respectively, $\bar{\boldsymbol{q}}(\boldsymbol{\theta})$ in place of $\boldsymbol{q}$. The role of a damping action within a compliance/impedance control scheme can be clearly understood when looking at VIDEO 133 and VIDEO 134.

Full-State Feedback

When the feedback law is based on a full set of measurements of the robot state, the transient performance of regulation control laws can be improved. Taking advantage of the availability of a joint torque sensor, we present a convenient design for the reduced model (11.9) of robots with elastic joints, including spring damping as a dissipation effect. Full-state feedback can be obtained in two phases, by combining a preliminary torque feedback with the motor feedback law in (11.28).

Using $\boldsymbol{\tau}_\mathrm{J} = \mathbf{K}(\boldsymbol{\theta} - \boldsymbol{q})$, the motor equation can be rewritten as

$$\mathbf{B}\ddot{\boldsymbol{\theta}} + \boldsymbol{\tau}_\mathrm{J} + \mathbf{D}\mathbf{K}^{-1}\dot{\boldsymbol{\tau}}_\mathrm{J} = \boldsymbol{\tau} \,.$$

The joint torque feedback

$$\boldsymbol{\tau} = \mathbf{B}\mathbf{B}_\theta^{-1}\boldsymbol{u} + \left(\mathbf{I} - \mathbf{B}\mathbf{B}_\theta^{-1}\right)\left(\boldsymbol{\tau}_\mathrm{J} + \mathbf{D}\mathbf{K}^{-1}\dot{\boldsymbol{\tau}}_\mathrm{J}\right) \,, \tag{11.39}$$

where $\boldsymbol{u}$ is an auxiliary input to be designed, transforms the motor equations into

$$\mathbf{B}_\theta\ddot{\boldsymbol{\theta}} + \boldsymbol{\tau}_\mathrm{J} + \mathbf{D}\mathbf{K}^{-1}\dot{\boldsymbol{\tau}}_\mathrm{J} = \boldsymbol{u} \,.$$

In this way, the apparent motor inertia can be reduced to a desired, arbitrary small value $\mathbf{B}_\theta$, with clear benefits in terms of vibration damping. For instance in the linear and scalar case, a very small B shifts the pair of complex poles in (11.23) to a very high frequency, with the joint behaving almost as a rigid one.

Setting now

$$\boldsymbol{u} = \mathbf{K}_{P,\theta}(\boldsymbol{\theta}_\mathrm{d} - \boldsymbol{\theta}) - \mathbf{K}_{D,\theta}\dot{\boldsymbol{\theta}} + \boldsymbol{g}(\boldsymbol{q}_\mathrm{d})$$

in (11.39) leads to the state feedback controller

$$\boldsymbol{\tau} = \mathbf{K}_P(\boldsymbol{\theta}_\mathrm{d} - \boldsymbol{\theta}) - \mathbf{K}_D\dot{\boldsymbol{\theta}} + \mathbf{K}_T[\boldsymbol{g}(\boldsymbol{q}_\mathrm{d}) - \boldsymbol{\tau}_\mathrm{J}] - \mathbf{K}_S\dot{\boldsymbol{\tau}}_\mathrm{J} + \boldsymbol{g}(\boldsymbol{q}_\mathrm{d}) \,, \tag{11.40}$$

with gains

$$\begin{aligned}
\mathbf{K}_P &= \mathbf{B}\mathbf{B}_\theta^{-1}\mathbf{K}_{P,\theta} \,, \\
\mathbf{K}_D &= \mathbf{B}\mathbf{B}_\theta^{-1}\mathbf{K}_{D,\theta} \,, \\
\mathbf{K}_T &= \mathbf{B}\mathbf{B}_\theta^{-1} - \mathbf{I} \\
\mathbf{K}_S &= \left(\mathbf{B}\mathbf{B}_\theta^{-1} - \mathbf{I}\right)\mathbf{D}\mathbf{K}^{-1} \,.
\end{aligned}$$

Indeed, the law (11.40) can also be rewritten in terms of $(\boldsymbol{\theta}, \boldsymbol{q}, \dot{\boldsymbol{\theta}}, \dot{\boldsymbol{q}})$, if the torque sensor is not available. However, keeping this structure of the gains preserves the interesting physical interpretation of what the full-state feedback controller is able to achieve.

11.1.4 Trajectory Tracking

As for rigid robots, the problem of tracking desired time-varying trajectories for robots with elastic joints is harder than that of achieving constant regulation. In general, solving this problem requires the use of full-state feedback and knowledge of all the terms in the dynamic model.

Under these conditions, we shall focus on the *feedback linearization* approach, namely a nonlinear state feedback law that leads to a closed-loop system with decoupled and exactly linear behavior for all the N joints of the robot (in fact, the link variables $\boldsymbol{q}$). The tracking errors along the reference trajectory are forced to be *globally exponentially stable*, with a decaying rate that can be directly specified through the choice of the scalar feedback gains in the controller. This fundamental result is the direct extension of the well-known *computed torque* method for rigid robots. Because of its relevant properties, the feedback linearization approach

can be used as a reference to assess the performance of any other trajectory tracking control law, which may possibly be designed using less/approximate model information and/or only partial-state feedback.

However, in the presence of joint elasticity, the design of a feedback linearization law is not straightforward. Furthermore, as soon as $\mathbf{S} \neq \mathbf{0}$, the dynamic model (11.7) will not satisfy the necessary conditions for exact linearization (nor those for input–output decoupling) when only a *static* (or, instantaneous) feedback law from the full state is allowed. Therefore, we will restrict our attention to the more tractable case of the reduced dynamic model (11.9) and sketch only briefly the more general picture.

As a second much simpler approach to trajectory tracking problems, we also present a *linear control design* that makes use of a model-based feedforward command and a precomputed state reference trajectory, as obtained from the inverse dynamics in Sect. 11.1.2, with the addition of a linear feedback from the full state. In this case, convergence to the desired trajectory is only locally guaranteed, i. e., the tracking errors should be small enough, but the control implementation is straightforward and the real-time computational burden is considerably reduced.

Feedback Linearization

Consider the reduced model (11.9) and let the desired motion be specified by a smooth reference trajectory $\boldsymbol{q}_{\mathrm{d}}(t)$ for the robot links. The control design will proceed by system inversion, in a similar way to the inverse dynamics computations of Sect. 11.1.2 but using now the current measures of the state variables $(\boldsymbol{q}, \boldsymbol{\theta}, \dot{\boldsymbol{q}}, \dot{\boldsymbol{\theta}})$ instead of the reference state evolution $(\boldsymbol{q}_{\mathrm{d}}, \boldsymbol{\theta}_{\mathrm{d}}, \dot{\boldsymbol{q}}_{\mathrm{d}}, \dot{\boldsymbol{\theta}}_{\mathrm{d}})$. Notably, there is no need to transform the robot equations into their state-space description, which is the standard form used in control design for general nonlinear systems; we will use the robot model directly in its second-order differential form (typical of mechanical systems).

The outcome of the inversion procedure will be the definition of a torque $\boldsymbol{\tau}$, in the form of a static state feedback control law, which cancels the original robot dynamics and replaces it with a desired linear and decoupled one of suitable differential order. In this sense, the control law *stiffens* the dynamics of the robot with elastic joints, irrespective of the fact that $\mathbf{K}$ is already relatively large (stiff joint) or small (soft joint). The feasibility of inverting the system from the chosen output $\boldsymbol{q}$ without causing instability problems (related to the presence of unobservable dynamics in the closed-loop system, after cancellation) is a relevant property of robots with elastic joints. In fact, this is the direct generalization to the nonlinear multiple-input multiple-output (MIMO) case of the possibility of inverting a scalar transfer function in the absence of zeros (see also the considerations made on (11.24)).

Rewrite the link equation in the compact form

$$\mathbf{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{n}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \mathbf{K}(\boldsymbol{q} - \boldsymbol{\theta}) = \mathbf{0}\,. \tag{11.41}$$

None of the above quantities depends instantaneously on the input torque $\boldsymbol{\tau}$. Therefore, we can differentiate once w.r.t. time and obtain

$$\mathbf{M}(\boldsymbol{q})\boldsymbol{q}^{[3]} + \dot{\mathbf{M}}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \dot{\boldsymbol{n}}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \mathbf{K}(\dot{\boldsymbol{q}} - \dot{\boldsymbol{\theta}}) = \mathbf{0}\,. \tag{11.42}$$

Proceeding one step further leads to

$$\begin{aligned}\mathbf{M}(\boldsymbol{q})\boldsymbol{q}^{[4]} + 2\dot{\mathbf{M}}(\boldsymbol{q})\boldsymbol{q}^{[3]} + \ddot{\mathbf{M}}(\boldsymbol{q})\ddot{\boldsymbol{q}} \\ + \ddot{\boldsymbol{n}}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \mathbf{K}(\ddot{\boldsymbol{q}} - \ddot{\boldsymbol{\theta}}) = \mathbf{0}\,,\end{aligned} \tag{11.43}$$

where $\ddot{\boldsymbol{\theta}}$ now appears. The motor acceleration is at the same differential level of $\boldsymbol{\tau}$ in the motor equation

$$\mathbf{B}\ddot{\boldsymbol{\theta}} + \mathbf{K}(\boldsymbol{\theta} - \boldsymbol{q}) = \boldsymbol{\tau}\,, \tag{11.44}$$

and thus, replacing $\ddot{\boldsymbol{\theta}}$ from (11.44), we get

$$\begin{aligned}&\mathbf{M}(\boldsymbol{q})\boldsymbol{q}^{[4]} + 2\dot{\mathbf{M}}(\boldsymbol{q})\boldsymbol{q}^{[3]} + \ddot{\mathbf{M}}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \ddot{\boldsymbol{n}}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \mathbf{K}\ddot{\boldsymbol{q}} \\ &= \mathbf{K}\mathbf{B}^{-1}\left[\boldsymbol{\tau} - \mathbf{K}(\boldsymbol{\theta} - \boldsymbol{q})\right]\,.\end{aligned} \tag{11.45}$$

We note that, using (11.41), the last term $\mathbf{K}(\boldsymbol{\theta} - \boldsymbol{q})$ in (11.45) may also be replaced by $\mathbf{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{n}(\boldsymbol{q}, \dot{\boldsymbol{q}})$.

Since the matrix $\mathbf{A}(\boldsymbol{q}) = \mathbf{M}^{-1}(\boldsymbol{q})\mathbf{K}\mathbf{B}^{-1}$ is always nonsingular, an arbitrary value $\boldsymbol{v}$ can be assigned to the fourth derivative of $\boldsymbol{q}$ by a suitable choice of the input torque $\boldsymbol{\tau}$. The matrix $\mathbf{A}(\boldsymbol{q})$ is the so-called decoupling matrix of the system and its nonsingularity is a necessary and sufficient condition for imposing a decoupled input–output behavior by nonlinear static state feedback. Moreover, (11.45) indicates that each component q_i of $\boldsymbol{q}$ needs to be differentiated $r_i = 4$ times in order to be algebraically related to the input torque $\boldsymbol{\tau}$ (r_i is the *relative degree* of q_i, when this is chosen as a system output). Since there are N link variables, the total sum of the relative degrees is $4N$, equal to the dimension of the state of a robot with elastic joints. All these facts taken together lead to the conclusion that, when inverting (11.45) to determine the input $\boldsymbol{\tau}$ that imposes $\boldsymbol{q}^{[4]} = \boldsymbol{v}$, there will be no dynamics left other than the one appearing in the closed-loop input–output map.

Therefore, choose

$$\begin{aligned}\boldsymbol{\tau} = \mathbf{B}\mathbf{K}^{-1}\left[\mathbf{M}(\boldsymbol{q})\boldsymbol{v} + \boldsymbol{\alpha}\left(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}, \boldsymbol{q}^{[3]}\right)\right] \\ + \left[\mathbf{M}(\boldsymbol{q}) + \mathbf{B}\right]\ddot{\boldsymbol{q}} + \boldsymbol{n}(\boldsymbol{q}, \dot{\boldsymbol{q}})\,,\end{aligned} \tag{11.46}$$

with

$$\boldsymbol{\alpha}(\boldsymbol{q},\dot{\boldsymbol{q}},\ddot{\boldsymbol{q}},\boldsymbol{q}^{[3]}) = \ddot{\mathbf{M}}(\boldsymbol{q})\ddot{\boldsymbol{q}} + 2\dot{\mathbf{M}}(\boldsymbol{q})\boldsymbol{q}^{[3]} + \ddot{\boldsymbol{n}}(\boldsymbol{q},\dot{\boldsymbol{q}}) \,,$$

where the terms in $\boldsymbol{\alpha}$ have been ordered according to the dependence on increasing orders of derivatives of $\boldsymbol{q}$. It is easy to verify that the control law (11.46) leads to a closed-loop system fully described by

$$\boldsymbol{q}^{[4]} = \boldsymbol{v} \,, \tag{11.47}$$

i. e., chains of four input–output integrators from each auxiliary input v_i to each link position output q_i, for $i = 1, \ldots, N$. Thus, the robot system has been exactly linearized and decoupled by the nonlinear feedback law (11.46). The improvement in performance can be appreciated in a comparative way by looking at VIDEO 135 and VIDEO 770.

The complete control law (11.46) is expressed as a function of the so-called *linearizing* coordinates $(\boldsymbol{q},\dot{\boldsymbol{q}},\ddot{\boldsymbol{q}},\boldsymbol{q}^{[3]})$ only. This has led to some misunderstandings in the past, as it seemed that the feedback linearization approach for robots with elastic joints would need direct measures of the link acceleration $\ddot{\boldsymbol{q}}$ and jerk $\boldsymbol{q}^{[3]}$, which are impossible to obtain with currently available sensors (or would require multiple numerical differentiations of position measures in real time, with critical noise problems).

When considering the latest technology in the field, it is now feasible to have a set of sensors for elastic joints measuring in a reliable and accurate way the motor position $\boldsymbol{\theta}$ (and, possibly, also its velocity $\dot{\boldsymbol{\theta}}$), the joint torque $\boldsymbol{\tau}_\mathrm{J} = \mathbf{K}(\boldsymbol{\theta} - \boldsymbol{q})$, as well as the link position $\boldsymbol{q}$. For instance, this is the arrangement of sensors available at each joint of the *LWR-III* lightweight manipulator, where expressly designed high-resolution incremental encoders of the magneto-resistive type are used for the motor position, joint torque sensing is based on a full bridge of strain gauges, and a high-end capacitive potentiometer is used for the absolute link position (Fig. 11.8). Therefore, only one numerical differentiation is needed in order to obtain a good estimate of $\dot{\boldsymbol{q}}$ and/or $\dot{\boldsymbol{\tau}}_\mathrm{J}$ as well. Note that, depending on the specific sensor resolution, it may also be convenient to evaluate $\boldsymbol{q}$ using the measures of $\boldsymbol{\theta}$ and $\boldsymbol{\tau}_\mathrm{J}$ as $\boldsymbol{\theta} - \mathbf{K}^{-1}\boldsymbol{\tau}_\mathrm{J}$.

With this in mind, it is easy to see that the following three sets of $4N$ variables

$$(\boldsymbol{q},\dot{\boldsymbol{q}},\ddot{\boldsymbol{q}},\boldsymbol{q}^{[3]}) \,, \qquad (\boldsymbol{q},\boldsymbol{\theta},\dot{\boldsymbol{q}},\dot{\boldsymbol{\theta}}) \,, \qquad (\boldsymbol{q},\boldsymbol{\tau}_\mathrm{J},\dot{\boldsymbol{q}},\dot{\boldsymbol{\tau}}_\mathrm{J})$$

are all equivalent state variables for a robot with elastic joints, related by globally invertible transformations. Therefore, under the assumption that the dynamic model is available, we can completely rewrite the feedback linearizing control law (11.46) in terms of the more *conventional* state $(\boldsymbol{q},\boldsymbol{\theta},\dot{\boldsymbol{q}},\dot{\boldsymbol{\theta}})$ or, taking advantage of a joint torque sensor, in terms of $(\boldsymbol{q},\boldsymbol{\tau}_\mathrm{J},\dot{\boldsymbol{q}},\dot{\boldsymbol{\tau}}_\mathrm{J})$. In particular, as a byproduct of (11.41) and (11.42), we have

$$\begin{aligned}\ddot{\boldsymbol{q}} &= \mathbf{M}^{-1}(\boldsymbol{q})\left[\mathbf{K}(\boldsymbol{\theta} - \boldsymbol{q}) - \boldsymbol{n}(\boldsymbol{q},\dot{\boldsymbol{q}})\right] \\ &= \mathbf{M}^{-1}(\boldsymbol{q})\left[\boldsymbol{\tau}_\mathrm{J} - \boldsymbol{n}(\boldsymbol{q},\dot{\boldsymbol{q}})\right] \,,\end{aligned} \tag{11.48}$$

and

$$\begin{aligned}\boldsymbol{q}^{[3]} &= \mathbf{M}^{-1}(\boldsymbol{q})\left[\mathbf{K}(\dot{\boldsymbol{\theta}} - \dot{\boldsymbol{q}}) - \dot{\mathbf{M}}(\boldsymbol{q})\ddot{\boldsymbol{q}} - \dot{\boldsymbol{n}}(\boldsymbol{q},\dot{\boldsymbol{q}})\right] \\ &= \mathbf{M}^{-1}(\boldsymbol{q})\left[\dot{\boldsymbol{\tau}}_\mathrm{J} - \dot{\mathbf{M}}(\boldsymbol{q})\ddot{\boldsymbol{q}} - \dot{\boldsymbol{n}}(\boldsymbol{q},\dot{\boldsymbol{q}})\right] \,,\end{aligned} \tag{11.49}$$

where the acceleration $\ddot{\boldsymbol{q}}$ appearing in (11.49) has already been computed through (11.48). Therefore, the exact linearizing and decoupling control law can be rewritten in terms of a static state feedback law of the form $\boldsymbol{\tau} = \boldsymbol{\tau}(\boldsymbol{q},\boldsymbol{\theta},\dot{\boldsymbol{q}},\dot{\boldsymbol{\theta}},\boldsymbol{v})$ or $\boldsymbol{\tau} = \boldsymbol{\tau}(\boldsymbol{q},\boldsymbol{\tau}_\mathrm{J},\dot{\boldsymbol{q}},\dot{\boldsymbol{\tau}}_\mathrm{J},\boldsymbol{v})$. Indeed, the evaluation of the various derivatives of the dynamic model terms that are present in these expressions should be properly organized or customized in order to optimize computations.

Based on the resulting (11.47), the trajectory-tracking problem is solved by setting

$$\begin{aligned}\boldsymbol{v} = \boldsymbol{q}_\mathrm{d}^{[4]} &+ \mathbf{K}_3\left(\boldsymbol{q}_\mathrm{d}^{[3]} - \boldsymbol{q}^{[3]}\right) + \mathbf{K}_2\left(\ddot{\boldsymbol{q}}_\mathrm{d} - \ddot{\boldsymbol{q}}\right) \\ &+ \mathbf{K}_1\left(\dot{\boldsymbol{q}}_\mathrm{d} - \dot{\boldsymbol{q}}\right) + \mathbf{K}_0\left(\boldsymbol{q}_\mathrm{d} - \boldsymbol{q}\right) \,,\end{aligned} \tag{11.50}$$

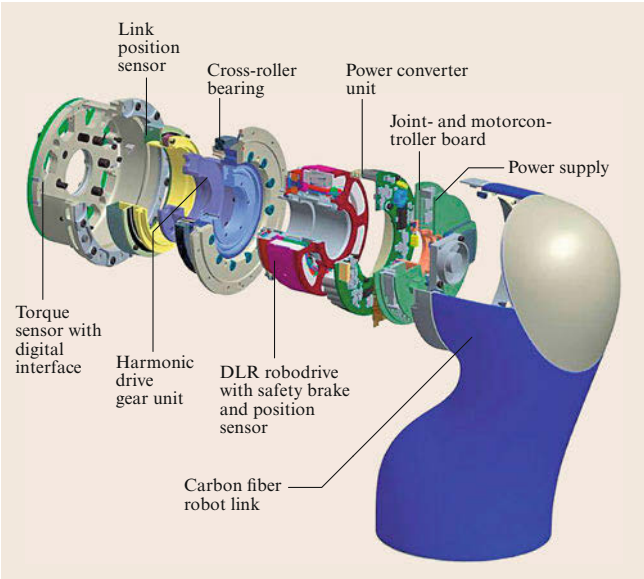


Fig. 11.8 Exploded view of a joint of the *DLR LWR-III* lightweight manipulator and its sensor suite

where it is assumed that the reference trajectory $\boldsymbol{q}_{\mathrm{d}}(t)$ is (at least) three times continuously differentiable (i. e., the fourth derivative $\boldsymbol{q}_{\mathrm{d}}^{[4]}$ exists), and the *diagonal* matrices $\mathbf{K}_0, \ldots, \mathbf{K}_3$ have scalar elements $\mathbf{K}_{\cdot,i}$ such that

$$s^4 + K_{3,i}s^3 + K_{2,i}s^2 + K_{1,i}s + K_{0,i}\,, \quad i = 1, \ldots, N\,,$$

are Hurwitz polynomials. In view of the obtained decoupling, the trajectory error

$$e_i(t) = q_{d,i}(t) - q_i(t)$$

for the i-th link satisfies then

$$e_i^{[4]} + K_{3,i}e_i^{[3]} + K_{2,i}\ddot{e}_i + K_{1,i}\dot{e}_i + K_{0,i}e_i = 0;\,,$$

yielding $e_i(t) \to 0$ in a global exponential way, for any initial state. A series of remarks are in order:

- When the initial state $(\boldsymbol{q}(0), \boldsymbol{\theta}(0), \dot{\boldsymbol{q}}(0), \dot{\boldsymbol{\theta}}(0))$ is matched with the reference trajectory and its first three derivatives at time $t = 0$ (for this check, (11.48) and (11.49) are to be used), exact reproduction of the reference trajectory is achieved at all times.
- In the case of a discontinuity of the reference position trajectory, or of any of its first three derivatives, at a time $t^* \in [0, T]$, a trajectory error occurs at $t = t^*$ that will again decay to zero, independently for each link and with the prescribed exponential rate.
- The choice of the gains $K_{3,i}, \ldots, K_{0,i}$ can be made by a pole placement technique (equivalent in this case to an eigenvalue assignment). Let $\lambda_1, \ldots, \lambda_4$ be four poles with negative real parts, possibly given in complex pairs and/or coincident, specifying the desired transient of the trajectory error. These closed-loop poles will be assigned by the unique choice of real and positive gains

$$\begin{aligned}
K_{3,i} &= -(\lambda_1 + \lambda_2 + \lambda_3 + \lambda_4)\,,\\
K_{2,i} &= \lambda_1(\lambda_2 + \lambda_3 + \lambda_4) + \lambda_2(\lambda_3 + \lambda_4) + \lambda_3\lambda_4\,,\\
K_{1,i} &= -\left[\lambda_1\lambda_2(\lambda_3 + \lambda_4) + \lambda_3\lambda_4(\lambda_1 + \lambda_2)\right]\,,\\
K_{0,i} &= \lambda_1\lambda_2\lambda_3\lambda_4\,.
\end{aligned}$$

  When the values of link and motor inertias are very different from each other, or when the joint stiffness is very large, the above fixed choice of gains has the drawback of generating too large control efforts. In those cases, a more tailored set of eigenvalues can be assigned scheduling their placement as a function of the physical data of robot inertias and joint stiffnesses.
- When compared to the *computed torque* method for rigid robots, the feedback linearization control for trajectory tracking given by (11.46)–(11.50) requires the inversion of the inertia matrix $\mathbf{M}(\boldsymbol{q})$ and the additional evaluation of derivatives of the inertia matrix and of other terms in the dynamic model.
- The feedback linearization approach can also be applied without any changes in the presence of viscous (or otherwise smooth) friction at the motor and link side. The inclusion of spring damping leads instead to a third-order decoupled differential relation between the auxiliary input $\boldsymbol{v}$ and $\boldsymbol{q}$, leaving thus a N-dimensional unobservable dynamics in the closed-loop system, which is, however, still asymptotically stable. In this case, only input–output (and not full-state) linearization and decoupling is achieved.

We conclude with some considerations on exact linearization/decoupling by feedback for the general model (11.7) of robots with elastic joints. Unfortunately, due to the presence of the inertial coupling matrix $\mathbf{S}(\boldsymbol{q})$ between links and rotors, the above control design can no longer be applied. Consider as an illustration the case of a constant $\mathbf{S}$ in (11.7), with the associated model simplifications. Solving for $\ddot{\boldsymbol{\theta}}$ from the motor equations, $\ddot{\boldsymbol{\theta}} = \mathbf{B}^{-1}\left(\boldsymbol{\tau} - \boldsymbol{\tau}_{\mathrm{J}} - \mathbf{S}^{\mathrm{T}}\ddot{\boldsymbol{q}}\right)$, and substituting into the link equations leads to

$$\begin{aligned}
\left[\mathbf{M}(\boldsymbol{q}) - \mathbf{S}\mathbf{B}^{-1}\mathbf{S}^{\mathrm{T}}\right]\ddot{\boldsymbol{q}} + \boldsymbol{n}(\boldsymbol{q}, \dot{\boldsymbol{q}}) - \left(\mathbf{I} + \mathbf{S}\mathbf{B}^{-1}\right)\boldsymbol{\tau}_{\mathrm{J}}\\
= -\mathbf{S}\mathbf{B}^{-1}\boldsymbol{\tau}\,.
\end{aligned}$$

In this case, the input torque $\boldsymbol{\tau}$ appears already in the expression for the link acceleration $\ddot{\boldsymbol{q}}$. Using (11.6), the expression of the decoupling matrix is then

$$\mathbf{A}(\boldsymbol{q}) = -\left[\mathbf{M}_{\mathrm{L}}(\boldsymbol{q}) + \mathbf{M}_{\mathrm{R}}(\boldsymbol{q})\right]^{-1}\mathbf{S}\mathbf{B}^{-1}\,,$$

which is never full rank in view of the structure (11.5) of $\mathbf{S}$. As a consequence, the necessary condition for obtaining (at least) input–output linearization and decoupling by static state feedback fails to hold.

However, by resorting to the use of a larger class of control laws, it is still possible to obtain an exact linearization and decoupling result. For this purpose, consider a *dynamic* state feedback controller of the form

$$\begin{aligned}
\boldsymbol{\tau} &= \boldsymbol{\alpha}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{\theta}, \dot{\boldsymbol{\theta}}, \boldsymbol{\xi}) + \boldsymbol{\beta}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{\theta}, \dot{\boldsymbol{\theta}}, \boldsymbol{\xi})\boldsymbol{v}\,,\\
\dot{\boldsymbol{\xi}} &= \boldsymbol{\gamma}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{\theta}, \dot{\boldsymbol{\theta}}, \boldsymbol{\xi}) + \boldsymbol{\delta}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{\theta}, \dot{\boldsymbol{\theta}}, \boldsymbol{\xi})\boldsymbol{v}\,, \qquad (11.51)
\end{aligned}$$

where $\boldsymbol{\xi} \in \mathbb{R}^{\nu}$ is the state of the dynamic compensator, $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$, $\boldsymbol{\gamma}$, and $\boldsymbol{\delta}$ are suitable nonlinear vector functions, and $\boldsymbol{v} \in \mathbb{R}^N$ is (as before) an external input used for trajectory-tracking purposes. It is possible to show in general that a dynamic compensator (11.51) of order

at most $\nu = 2N(N-1)$ can be designed so as to yield a closed-loop system that is globally described by

$$\boldsymbol{q}^{[2(N+1)]} = \boldsymbol{v} \,, \tag{11.52}$$

in place of (11.47). Note the coincidence of this differential order with the one found for exact reproducibility of a desired trajectory $\boldsymbol{q}_\mathrm{d}(t)$ in Sect. 11.1.2. The tracking problem is then solved by working on (11.52), with a direct generalization of the linear stabilization design in (11.50).

It is interesting to give a physical interpretation for this control result. The structural obstruction to input–output decoupling of the model (11.7) is related to the fact that the motion of a link attached to an elastic joint is affected *too soon* (at the second differential level) by torques originating at other elastic joints. This is due to the inertial couplings present between motors and links. The addition of dynamics (i. e., of integrators) in the controller slows down these low-energy paths and allows for the high-energy effects of the elastic torque at the local joint (a fourth-order differential path) to come into play. This dynamic balancing allows both input–output decoupling and exact linearization of the extended system (having the robot and the controller states).

Linear Control Design

The feedback linearization approach to trajectory tracking gives rise to a rather complex nonlinear control law. Its main advantage of globally enforcing a linear and decoupled behavior to the trajectory error dynamics should be traded-off with a control design that achieves only local stability around the reference trajectory, but is much simpler to implement (and may run at higher sampling rates).

For this, the inverse dynamics results of Sect. 11.1.2 can be used, assuming either the reduced or the complete robot model, with and without dissipative terms. Given a sufficiently smooth desired link trajectory $\boldsymbol{q}_\mathrm{d}(t)$, we have seen that it is always possible to associate:

1. The nominal torque $\boldsymbol{\tau}_\mathrm{d}(t)$ needed for its exact reproduction.
2. The reference evolution of all other state variables (e.g., $\boldsymbol{\theta}_\mathrm{d}(t)$, as given by (11.13), or $\boldsymbol{\tau}_{\mathrm{J},d}(t)$).

These signals define a sort of steady-state (though, time-varying) operation for the system.

A simpler tracking controller combines a model-based feedforward term with a linear feedback term using the trajectory error. The linear feedback locally stabilizes the system around the reference state trajectory, whereas the feedforward torque is responsible for maintaining the robot along the desired motion as soon as the error has vanished (see some examples on the tracking of square paths in VIDEO 136).

Using full-state feedback, two possible controllers of this kind are

$$\begin{aligned}\boldsymbol{\tau} = \boldsymbol{\tau}_\mathrm{d} &+ \mathbf{K}_{P,\theta}(\boldsymbol{\theta}_\mathrm{d} - \boldsymbol{\theta}) + \mathbf{K}_{D,\theta}(\dot{\boldsymbol{\theta}}_\mathrm{d} - \dot{\boldsymbol{\theta}}) \\ &+ \mathbf{K}_{P,q}(\boldsymbol{q}_\mathrm{d} - \boldsymbol{q}) + \mathbf{K}_{D,q}(\dot{\boldsymbol{q}}_\mathrm{d} - \dot{\boldsymbol{q}})\end{aligned} \tag{11.53}$$

and

$$\begin{aligned}\boldsymbol{\tau} = \boldsymbol{\tau}_\mathrm{d} &+ \mathbf{K}_{P,\theta}(\boldsymbol{\theta}_\mathrm{d} - \boldsymbol{\theta}) + \mathbf{K}_{D,\theta}(\dot{\boldsymbol{\theta}}_\mathrm{d} - \dot{\boldsymbol{\theta}}) \\ &+ \mathbf{K}_{P,J}(\boldsymbol{\tau}_{\mathrm{J,d}} - \boldsymbol{\tau}_\mathrm{J}) + \mathbf{K}_{D,J}(\dot{\boldsymbol{\tau}}_{\mathrm{J,d}} - \dot{\boldsymbol{\tau}}_\mathrm{J}) \,.\end{aligned} \tag{11.54}$$

These trajectory tracking schemes are the most common in the control practice for robots with elastic joints. In the absence of full-state measurements, they can be combined with an observer of the unmeasurable quantities. An even simpler realization is

$$\boldsymbol{\tau} = \boldsymbol{\tau}_\mathrm{d} + \mathbf{K}_P(\boldsymbol{\theta}_\mathrm{d} - \boldsymbol{\theta}) + \mathbf{K}_D(\dot{\boldsymbol{\theta}}_\mathrm{d} - \dot{\boldsymbol{\theta}}) \,, \tag{11.55}$$

which uses only motor measurements and relies on the results obtained for the regulation case.

The different gain matrices used in (11.53)–(11.55) have to be tuned using a linear approximation of the robot system. This approximation may be obtained at a fixed equilibrium point or around the actual reference trajectory, leading respectively to a linear time-invariant or to a linear time-varying system. While the existence of (possibly time-varying) stabilizing feedback matrices is guaranteed by the controllability of these linear approximations, the validity of the approach is only *local* in nature and the region of convergence will depend both on the given trajectory and on the robustness of the designed linear feedback.

It should be mentioned that such a control approach to trajectory tracking problems can be used also for robots with flexible links. Once the inverse dynamics problem has been solved (for the case of link flexibility, this typically results in a noncausal solution), a controller of the form (11.55) (or (11.53), with link deflection $\boldsymbol{\delta}$ and deflection rate $\dot{\boldsymbol{\delta}}$ replacing, respectively, $\boldsymbol{q}$ and $\dot{\boldsymbol{q}}$), can be directly applied.

11.1.5 Further Reading

This section contains the main references for the part of this chapter on robots with joint flexibility. In addition, we point out to a larger bibliography for topics that have not been included here.

Early interest in problems arising from the presence of flexible transmissions in industrial robots dates back to [11.1, 2], with first experimental findings on the

Avidbots Ex1016
Page 305 of 2154

GE P-50 arm. Relevant mechanical considerations involved in the design of robot arms and in the evaluation of their compliant elements can be found in [11.3].

One of the first studies on the inclusion of joint elasticity in the dynamic modeling of robot arms is due to [11.4]. The detailed analysis of the model structure presented in Sect. 11.1.1 comes from [11.5], with some updates from [11.6]. The simplifying assumption leading to the reduced model was introduced in [11.7]. The special case of motors mounted on the driven links is considered in [11.8]. The observation that joints with limited elasticity lead to a singularly perturbed dynamic model was first recognized in [11.9]. Symbolic manipulation programs for the automatic generation of dynamic models of robots with elastic joints were also developed very early [11.10].

The inverse dynamics computations in Sect. 11.1.2 can be traced back to [11.11, 12]. The use of programs such as Modelica for the numerical evaluation of inverse dynamics is highlighted in [11.6]. An efficient Newton–Euler inverse dynamics algorithm for robots with elastic joints has been proposed in [11.12].

The first specific control designs were based on decentralized linear controllers, see, e.g., [11.13] and [11.14], where a fourth-order dynamics local to each elastic joint was used. However, schemes with proved global convergence characteristics appeared only later.

In Sect. 11.1.3, the PD controller with constant gravity compensation is a contribution of [11.15]. This was also extended to the case of robots with flexible links in [11.16]. The first two versions of regulation control with online gravity compensation are due, respectively, to [11.17] and [11.18]. Both these control laws were extended to Cartesian compliance schemes in [11.19–21]. The PD-like controller (11.38), with perfect cancellation of gravity effects on link motion, is a result of [11.22]. A general regulation framework based on energy shaping has been proposed in [11.23]. In the absence of any information on gravity, a proportional–integral–derivative (PID) regulator with semiglobal stability properties has been presented in [11.24], while a global learning scheme was proposed in [11.25].

The presentation of a full-state feedback design for regulation (and tracking) follows the idea used in [11.26]. Special interest has been devoted over the years to joint torque feedback, from the placement of torque sensors on transmission shafts [11.27] or within harmonic drives [11.28] to their use for achieving robust control performance [11.29].

The fact that the reduced model of robots with elastic joints can always be feedback linearized via static state feedback was shown for the first time in [11.7]. A previous similar result for a specific robot kinematics can be found in [11.30]. The discrete-time implementation of the feedback linearization approach has been studied in [11.31], while its robustness properties were analyzed in [11.32]. Feedback linearization and input–output decoupling were also considered for the case of viscoelastic joints in [11.33], and for robots with joints of mixed type, some rigid and some elastic, in [11.34]. The same idea of inversion control, leading to input–output decoupling and linearization, has also been successfully applied to the tracking problem of joint-level trajectories in robots with flexible links [11.35].

For the general model of robots with elastic joints, the use of dynamic state feedback was first proposed in [11.36]. A comparative study of the errors induced by neglecting the motor–link inertia couplings (thus, for robots that are not linearizable by static state feedback) was carried out in [11.37]. The general algorithm for constructing the dynamic linearizing feedback is presented in [11.38, 39].

The benefit of combining a stabilizing feedback with a feedforward command in robots with flexible joints was experimentally shown for the first time in [11.39] (see also the VIDEO 770) and then formalized in [11.11].

Other control design methodologies available for trajectory tracking, which were not discussed in Sect. 11.1.4, are based on singular perturbation, backstepping, or passivity. Nonlinear controllers based on the two-time scale separation property were proposed by [11.40] and [11.41]. These corrective controllers are an outcome of the singular perturbation model form and should be preferred in the case of very stiff joints, as they do not generate high-gain laws in the limit. Backstepping is based on the idea of considering the joint elastic torque [11.42] or the motor position [11.43] as an intermediate fictitious input to be used to control the link equations, and then to design the actual torque input in the motor equations so as to follow the reference behavior for the previous intermediate input. The main advantage is that this design may be easier to be transformed into an adaptive version. Adaptive control results for robots with elastic joints include the high-gain (approximate) schemes [11.44] and [11.45], as well as the global (but very complex) solution obtained in [11.46], both analyzed using only the reduced dynamic model. Moreover, robust control schemes have been proposed in [11.47], based on sliding mode techniques, and in [11.48–52], using iterative learning on repetitive tasks.

As for optimal control results, the intrinsic fourth-order dynamics associated to flexible joints has barred so far the derivation of analytical or numerically efficient results. Finding the time-optimal velocity profile

Avidbots Ex1016
Page 306 of 2154

on a constrained geometric path is a solved problem in the rigid case, but only an approximate solution is available for robots with elastic joints [11.49]. A useful generalization from the rigid to the elastic case is the dynamic trajectory scaling algorithm of [11.50], which provides a closed-form expression to the uniform time scaling that recovers feasibility of commanded torques w.r.t. their bounds. Selected optimal control problems for single viscoelastic joints, involving either the maximization of stored potential energy or the minimization of rest-to-rest motion time, have been tackled and solved in [11.51] and [11.52], respectively.

Different state observers have been presented, starting from an approximate one [11.53] up to the exact ones in [11.5, 54], in which a tracking controller based on the estimated state was also tested. In all cases, link position or link position and velocity measurements are assumed. On the other hand, the use of motor position and link acceleration has been shown to be a feasible alternative for robust state reconstruction [11.55].

Finally, the force control problem in constrained tasks for robots with elastic joints has been tackled following a singular perturbation technique [11.56, 57], an inverse dynamics approach [11.58], or an adaptive strategy [11.59].

A unified passivity-based approach has been presented in [11.60], where an inner torque feedback loop is incorporated for shaping (viz., reducing) the motor inertia. This enables approximate gravity compensation (as done in Sect. 11.1.3), as well as to assign a desired Cartesian stiffness relation for the link positions, mimicking the impedance control of the rigid case for handling interactions and contacts with the environment.

11.2 Robots with Flexible Links

Arm flexibility is a dynamic behavior in which kinetic energy interacts with elastic potential energy. Kinetic energy is stored in moving inertia, and potential energy is stored in compliant members. Flexible links provide both the inertia and compliance distributed throughout the member. Flexibility manifests itself as mechanical oscillations and static deflections, greatly complicating the motion control of a mechanical arm. If the time to settle the oscillations is significant relative to the cycle time of the overall task, flexibility will be a major consideration in the arm design.

11.2.1 Design Issues

We will deal with these issues in a modular way, separating link flexibility and joint flexibility. The first topic of link flexibility deals with the inherent spatial distribution of compliance and inertia, perhaps with both effects existing in the same element of material. The links of a manipulator arm can usually be stiffened by adding structural mass or by improving the material properties or distribution over the links. At some points, the links may be treated as rigid. The designer must realize that an arm with heavy rigid links and flexible joints may be more flexible dynamically than a lighter arm with more link flexibility and less inertia. The consequences are predicted by the natural frequency of the first flexible mode of motion, with the joint actuators locked, which can be approximated by

$$\omega_1 = \sqrt{\frac{k_{\text{eff}}}{I_{\text{eff}}}}\,, \tag{11.56}$$

where k_{eff} is the effective spring constant and I_{eff} is the effective inertia. For the elastic joint case, k_{eff} and I_{eff} are directly available from the elastic joint parameters and mass properties of the arm. The *effective* values applicable to a simple beam in bending, for example, would result from boundary conditions that are clamped at one end and free at the other and would be selected to yield a good approximation to the natural frequency. This natural frequency is readily available from handbooks on vibration in this case as $\omega_1 = 3.52\sqrt{EI/ml^4}$. (The variables are used in (11.57) and (11.58) and defined below.) More complex geometries can be evaluated by various means, including the transfer matrix method discussed in Sect. 11.2.2 on modeling. The arm should be considered flexible when this frequency is low enough to interfere with the design of controllers based on rigid assumptions. The common PD control of a single joint on a flexible link can only achieve adequate damping of a closed-loop design when the magnitude of the closed loop poles is less than about $1/3$ of ω_1 [11.61].

A distillation of the key aspects of the mechanics of materials is now in order. In some cases static compliance effects represent a member sufficiently, whereas in other cases the true distributed nature of compliance and mass and its dynamics must be modeled. A long structural member with forces f and moments M perpendicular to its long axis is subject to bending. Static bending relates the moment at the axial location x to the displacement w

$$M(x) = EI(x)\frac{\partial^2 w}{\partial x^2}\,, \tag{11.57}$$

where E is the elastic modulus of the material, and $I(x)$ is the area moment of inertia about the neutral axis of the cross section. The deflection at any point is obtained by integrating this equation from a reference point to a desired point, such as the end of the link.

This is a description of compliance that can be used when mass is isolated from elasticity. If mass is distributed throughout the beam with material mass density per unit volume ρ, the time t must be incorporated so that

$$m(x)\frac{\partial^2 w(x,t)}{\partial t^2}+\frac{\partial^2}{\partial x^2}\left(EI(x)\frac{\partial^2 w(x,t)}{\partial x^2}\right)=0\,. \tag{11.58}$$

Here $m(x)$ is the mass density per unit length, incorporating the material properties and the cross-sectional area at x. Assumptions implicit in this equation, which is called the Bernoulli–Euler equation, include the minimal impact of shear distortion and rotational inertia of the cross section of the beam, which is valid for long beams. The Timoshenko beam model relaxes these assumptions but is not included here.

Torsional deflection of an angle θ results if a twisting moment T about the long beam axis occurs, such that, in the static case

$$\theta=\frac{Tl}{JG}=\alpha_{\theta T}T\,, \tag{11.59}$$

where G is the shear modulus of the material, and J is the polar area moment of inertia about the neutral axis of the beam. Again, the addition of a distributed mass and considering the dynamics produces a partial differential equation with independent variables in space and time

$$\mu(x)\frac{\partial^2\theta}{\partial t^2}=GJ\frac{\partial^2\theta}{\partial x^2}\,, \tag{11.60}$$

where $\mu(x)$ is the rotary mass moment of inertia per unit length of the shaft. While (11.60) is potentially appropriate for links, the smaller rotational mass moment of inertia of a long slender shaft relative to the links it drives results in the static compliance effects of torsion given by (11.59) being of most importance.

Tension and compression effects should also be acknowledged, although they tend to be the least significant. Here deflection δ in the x-direction (the long axis of the member) results from axial force F_a and is calculated as $\delta=\frac{F_aL}{AE}$

$$\delta=\frac{F_{\mathrm{a}}L}{AE}=\alpha_{XF}F_{\mathrm{a}}\,, \tag{11.61}$$

where L is the element length and A is its cross sectional area. In the dynamic case, with mass density ρ and displacement ξ, the axial motion is described by

$$\rho\frac{\partial^2\xi}{\partial t^2}=E\frac{\partial^2\xi}{\partial x^2}\,. \tag{11.62}$$

Note that these effects are special cases of the more general elastic behavior of a member subject to acceleration and external loading, specialized to the member with one long axis. The same general phenomena apply to joint structures (e.g., bearings and couplings) and drive-train components (e.g., gears and cables). The functional constraints on the structural design of these members may be more severe and limit the designer's ability to make them more rigid by adding more material. Under this constraint, the combined compliance of the flexible joint design and the rigidized link design results in inferior performance of the combination. A less rigid link with less inertia will raise the lowest natural frequency. Thus the design of the link and the joint must be coordinated.

Flexibility is only one of several link design constraints. Static compliance is obviously another based on the above discussion. Buckling and strength are two more considerations that constrain some of the same design parameters. While it has been shown that flexibility is a dominant constraint for typical arm design regimes [11.62], a simple optimization of bending stiffness of a beam with a tubular geometry illustrates the fallacy of ignoring buckling. If the radius of the tube is varied to maximize bending stiffness with constrained total mass, the radius increases without bound, producing a thin shell of infinitesimal thickness and infinite radius that lacks robustness. The true constraint becomes local buckling of the tube under slight perturbations of the applied load. A thin metal can illustrates this behavior as it crushes. Strength is another applicable constraint. Various materials differ in their strength, typically represented by their elastic limit or endurance limit on the stress at a point in the member. A sufficiently stiff member may fail due to stress, particularly at points of stress concentration.

In this presentation of options to overcome flexibility, and link flexibility in particular, radically different operational and design strategies should be recognized even though they are not the core of the treatment. The elastic modulus relates stress to strain, but physically the strain rate is sometimes a relevant term, providing structural damping to augment the joint or control-based damping. Composite materials have substantially more damping inherent in their behavior. Damping can be enhanced with passive damping treatments, with constrained layer damping having particularly pronounced effects [11.63]. Smart materials can also be

configured to enhance damping [11.64]. An operational strategy known as *bracing* can be combined with redundant actuation to achieve a large workspace for a flexible arm that provides gross motion and a smaller precision workspace after docking the large arm with a stationary structure, as described in [11.65].

The implication above is that flexibility was always due to the moving parts of the arm. However, the arm base itself may be a source of significant flexibility. If the base is stationary, the analysis may be almost identical to that of the flexible links of an arm. However, if the arm is mounted on a vehicle, tires or tracks may be the source of flexibility and empirical evaluation of these new elements might be the best solution. The vehicle could be a boat, aircraft, or space vehicle. Here there is no elastic connection to an inertial reference frame and some fundamental changes in approach will be needed. These changes are not unexplored and in some cases are simple extensions of the following approaches.

11.2.2 Modeling of Flexible Link Arms

Mathematical models of link flexibility trade accuracy for ease of use. It is therefore key to determine the effects that are important to represent accurately. The techniques presented here will assume linear elasticity with light damping. Rotational motions must be of modest angular rate so that centrifugal stiffening can be ignored. Small deflections will generally be assumed. These are reasonable assumptions for most robotic devices, but can be violated in more exotic applications, requiring reevaluation of the assumptions and more tortuous modeling approaches.

Four model types will be examined:

1. Lumped-element models having compliance or inertia but not both in a given member.
2. Finite-element models, which will be only briefly addressed.
3. Assumed mode models which readily include nonlinear dynamic behavior.
4. Transfer matrix models that incorporate the true distributed intermingling of compliance and inertia and consequently the infinite-dimensional nature of flexible link arms.

With the limited space available here, the discussion will be preliminary but should enable the reader to pursue one or more alternatives knowledgably.

For lumped elements, one may build on the rigid kinematics presented in the section on fundamentals of kinematics (Chap. 2) that establishes rigid transformation matrices $\mathbf{A}_i$. The 4×4 homogenous transformations that describe position can be used to describe deflection as well. Assuming small motions and static behavior (or negligible member mass), the transformation produced by elastic bending, torsion, and compression is

$$\mathbf{E}_i = \begin{pmatrix} 1 & -\alpha_{\theta Fi}F^i_{Yi} - \alpha_{\theta Mi}M^i_{Zi} \\ \alpha_{\theta Fi}F^i_{Yi} + \alpha_{\theta Mi}M^i_{Zi} & 1 \\ \alpha_{\theta Fi}F^i_{Zi} - \alpha_{\theta Mi}M^i_{Yi} & \alpha_{Ti}M^i_{Xi} \\ 0 & 0 \\ -\alpha_{\theta Fi}F^i_{Zi} + \alpha_{\theta Mi}M^i_{Yi} & \alpha_{Ci}F^i_{Xi} \\ -\alpha_{Ti}M_{Xi} & \alpha_{XFi}F^i_{Yi} + \alpha_{XMi}M^i_{Zi} \\ 1 & \alpha_{XFi}F^i_{Zi} + \alpha_{XMi}M^i_{Yi} \\ 0 & 1 \end{pmatrix}, \tag{11.63}$$

where

α_{Ci} = coefficient in compression, displacement/force ,

α_{Ti} = coefficient in torsion, angle/moment ,

$\alpha_{\theta Fi}$ = coefficient in bending, angle/force ,

$\alpha_{\theta Mi}$ = coefficient in bending, angle/moment ,

α_{XFi} = coefficient in bending, displacement/force ,

α_{XMi} = coefficient in bending, displacement/moment ,

F^j_{Xi} = the force at the end of link i in the X direction of coordinate frame j ,

F^j_{Yi} = the force at the end of link i in the Y direction of coordinate frame j ,

F^j_{Zi} = the force at the end of link i in the Z direction of coordinate frame j ,

M^j_{Xi} = the moment at the end of link i in the X direction of coordinate frame j ,

M^j_{Yi} = the moment at the end of link i in the Y direction of coordinate frame j ,

M^j_{Zi} = the moment at the end of link i in the Z direction of coordinate frame j .

The listed coefficients depend on the construction of the element and are readily found for a slender beam from simple strength of materials. In other cases, a finite-element model or empirical measurements may be more readily used.

Alternating transformations for the undeformed and deformed aspects of N members (links or joints) produces the position vector of the end of arm (EOA)

$$\boldsymbol{p}^0 = (\mathbf{A}_1\mathbf{E}_1\mathbf{A}_2\mathbf{E}_2 \ldots \mathbf{A}_i\mathbf{E}_i\mathbf{A}_{i+1}\mathbf{E}_{i+1} \ldots \mathbf{A}_N\mathbf{E}_N)\begin{pmatrix}0\\0\\0\\1\end{pmatrix}. \tag{11.64}$$

If the massless elements connect rigid lumped masses, a linear spatial model is readily obtained, as described in [11.66]. Servo-controlled joints can also be inserted into this model. This analysis proceeds by examining the deflection of the elastic members produced by joint motion and the resulting forces and moments on the rigid inertias at the ends of the chain of links. The reference concentrates on the special case of only two inertias at each end of the chain but the technique is easily extended to inertias in the middle of the chain as well. The results are six linear second-order equations for each inertia and one first-order equation for each joint.

If the compliance and inertia properties are both to be treated as distributed throughout the same element, the partial differential equations (PDEs) introduced in the previous section for bending, torsion, and compression must be employed. Still insisting on a linear treatment, the general solution of these equations can be found in terms of each element with adjoining elements, i. e., the boundary conditions on the PDE. The convenient approach of Laplace transformation of these linear equations into the frequency domain, followed by factoring out the boundary conditions into a matrix–vector product, results in a technique known as the transfer matrix method (TMM) [11.67]. This method has been applied productively to general elastomechanics problems as well as specifically to flexible arms [11.68].

The transfer matrix relates variables at two stations along the arm with the number of variables depending on the model complexity. For the planar bending, deflection of the beam is shown in Fig. 11.9

$$z_1 = \begin{pmatrix} -W \\ \Psi \\ M \\ V \end{pmatrix}_1 = \begin{pmatrix} -\text{displacement} \\ \text{angle} \\ \text{moment} \\ \text{shear force} \end{pmatrix}_{\text{at station 1}} ,$$
$$z_0 = \mathbf{T} z_1 , \qquad (11.65)$$

where T is the appropriate element transfer matrix.

If the element is a simple rotary spring with constant k and a damper with constant b, then the angle of rotation θ is related to the moment by a differential equation with the following Laplace-domain representation

$$M = -L(k\theta + b\dot{\theta}) \Rightarrow k\Theta(s) + bs\Theta(s) = M_0 = M_1 ,$$
$$W_0 = W_1 ,$$
$$V_0 = V_1 ,$$
$$\Psi_0 = \Psi_1 - \Theta(s) = \Psi_1 - \frac{1}{k+bs} M_1 .$$

Note that the only variable changed across the element is the angle related to the moment. The transfer matrix representation conventionally uses zero initial conditions and converts to the Fourier representation with $s = \imath\omega$. This leads to the transfer function

$$\mathbf{T} = \mathbf{C}(\imath\omega) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & \frac{1}{k+b(\imath\omega)} & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} ,$$

which has the same form of the joint controller as shown in (11.69).

The beam model is much more complex, and the Euler–Bernoulli model gives

$$\mathbf{B} = \begin{pmatrix} c_0 & lc_1 & ac_2 & alc_3 \\ \frac{\beta^4 c_3}{l} & c_0 & \frac{ac_1}{l} & ac_2 \\ \frac{\beta^4 c_2}{a} & \frac{\beta^4 l c_3}{a} & c_0 & lc_1 \\ \frac{\beta^4 c_1}{al} & \frac{\beta^4 c_2}{a} & \frac{\beta^4 c_3}{l} & c_0 \end{pmatrix} , \qquad (11.66)$$

where

$$\beta^4 = \omega^2 l^4 \mu/(EI) ; \quad a = l^2/(EI) ,$$
$$c_0 = (\cosh\beta + \cos\beta)/2 ,$$
$$c_1 = (\sinh\beta + \sin\beta)/(2\beta) ,$$
$$c_2 = (\cosh\beta - \cos\beta)/(2\beta^2) ,$$
$$c_3 = (\sinh\beta - \sin\beta)/(2\beta^3) ,$$
$$\mu = \text{ density/unit length} ,$$
$$\omega = \text{ circular frequency of vibration} ,$$
$$E = \text{ elastic modulus} ,$$
$$I = \text{ cross-sectional area moment of inertia} .$$

The spatial variable has been transformed to the spatial Laplace variable but no longer appears explicitly. The Laplace time variable remains and is designated by s or by the frequency variable $\omega = -\imath s$, where $\imath = \sqrt{-1}$.

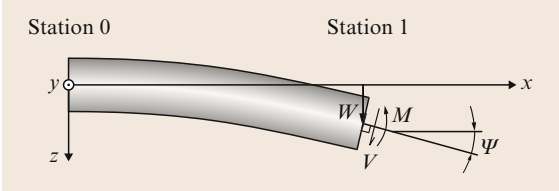


Fig. 11.9 State vector for the bending transfer matrix

The other transfer matrices needed for a simple planar model are as follows.

For a rotation in the plane of angle φ

$$\mathbf{A} = \begin{pmatrix} \frac{1}{\cos\varphi} & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ m_s\omega^2 \sin\varphi \tan\varphi & 0 & 0 & \cos\varphi \end{pmatrix}, \quad (11.67)$$

where m_s is the sum of all outboard masses from the angle to the end of the arm. This is an approximation resulting from ignoring compression elasticity. These elements appear as an additional mass to the extent they are translated with beam compression.

For a rigid mass, simply applying Newton's laws and collecting term yields

$$\mathbf{R} = \begin{pmatrix} 1 & l & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -\frac{m\omega^2 l}{2} & I_m\omega^2 - \frac{m\omega^2 l^2}{2} & 1 & l \\ m\omega^2 & -\frac{m\omega^2 l}{2} & 0 & 1 \end{pmatrix}, \quad (11.68)$$

where m is the mass of the body, I_m is the mass moment of inertia about an axis through the center of mass and perpendicular to the plane of the arm, l is the length of the mass (the distance between the points of attachment at stations i and $i+1$), and $l/2$ is the distance to the center of mass from station i.

For a controlled joint with the controller transfer function (joint torque/joint angle) $= k(\imath\omega)$

$$\mathbf{C} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & \frac{1}{k(\imath\omega)} & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}. \quad (11.69)$$

When combined to represent the planar arm with two joints as shown in Fig. 11.10, this analysis yields a composite transfer matrix that relates the four state variables at the two ends of the arm. Further analysis results from imposing the known boundary conditions at these ends.

The transfer matrix representation of an arm presented in [11.69] and pictured in Fig. 11.10 is shown in (11.70).

$$\begin{pmatrix} -W \\ \Psi \\ M \\ V \end{pmatrix}_0 = \mathbf{R}_1\mathbf{B}_2\mathbf{R}_3\mathbf{A}_4\mathbf{C}_5\mathbf{B}_6\mathbf{R}_7 \begin{pmatrix} -W \\ \Psi \\ M \\ V \end{pmatrix}_7 . \quad (11.70)$$

In the example, the left end is pinned and the right end is free, yielding

$$z_0 = \begin{pmatrix} -W \\ \Psi \\ M \\ V \end{pmatrix}_0 = \begin{pmatrix} u_{11} & u_{12} & u_{13} & u_{14} \\ u_{21} & u_{22} & u_{23} & u_{24} \\ u_{31} & u_{32} & u_{33} & u_{34} \\ u_{41} & u_{42} & u_{43} & u_{44} \end{pmatrix} \begin{pmatrix} -W \\ \Psi \\ M \\ V \end{pmatrix}_7 ,$$

$$\begin{pmatrix} u_{11} & u_{12} \\ u_{31} & u_{32} \end{pmatrix} \begin{pmatrix} -W \\ \Psi \end{pmatrix}_7 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}. \quad (11.71)$$

The determinant of the 2×2 matrix must be zero when ω is a natural frequency, or a system eigenvalue if it is complex. This can be found by a numerical search. Evaluating variables (position, angle, shear, and moment) along the length of the arm with the eigenvalues used for the values of ω yields the eigenfunctions. If boundary conditions are not zero but known for given frequencies, the frequency response of the system can be obtained. Hence Bode plots are readily obtained for boundary condition forcing. Interior forcing can be handled using an extended state vector as described in [11.67], which was applied to arms in [11.61] and most recently updated with modern programming techniques in [11.68]. Inverse fast Fourier transform (FFT) will produce a time-domain response from the frequency response.

The creation of a state-space model in the time domain is attractive because it enables the use of powerful state-space design techniques and is compatible with the nonlinear behavior of arms undergoing large motion and experiencing centrifugal and Coriolis forces. The assumed modes method allows such a model to be constructed efficiently. The presentation here follows the early introduction of a recursive manner to calculate these equations introduced in [11.70]. The flexible kinematics of the links must be expressed as a sum of basis functions also known as assumed modes (shapes) $\varphi_i(x)$ with time-variable amplitudes $\delta_i(t)$

$$w(x,t) = \sum_{i=1}^{\infty} \delta_i(t)\varphi_i(x) . \quad (11.72)$$

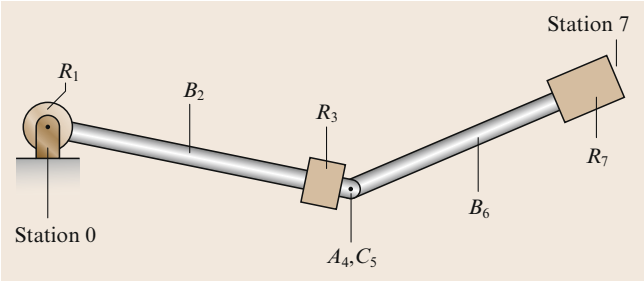


Fig. 11.10 Transfer matrix representation of an arm

The amplitudes and their derivatives become the states of the model. Compatible joint angle variables and their derivatives are also included as the rigid state variables. Joint angles tangent to the ends of the flexible links are compatible with mode shapes based on the clamped boundary conditions at one end and free boundary conditions at the other. This choice is generally chosen for simulation and the rigid coordinates are directly measured by standard joint angle transducers. If rigid coordinates are angles measured between lines connecting successive joint axes, the end of arm is known in terms of rigid coordinates alone and pinned–pinned boundary conditions are consistent. This produces advantages for inverse dynamics calculations as seen shortly.

The flexible and rigid kinematics combined describes the position and velocity of every point on the arm and can be used to express the kinetic energy $\mathcal{T}$ and the potential energy $\mathcal{V}$. These expressions are used in the conservative form of Lagrange's equation

$$\frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial \mathcal{T}}{\partial \dot{q}_i} - \frac{\partial \mathcal{T}}{\partial q_i} + \frac{\partial \mathcal{V}}{\partial q_i} = F_i \,, \tag{11.73}$$

where F_i is the force that does work as q_i is varied.

The kinematics of the deflection are again described by 4×4 transformation matrices, but this time the assumed mode shapes j are summed to give the position of link i

$$\boldsymbol{h}_i^i(\eta) = \begin{pmatrix}\eta\\0\\0\\1\end{pmatrix} + \sum_{j=i}^{m_i}\delta_{ij}\begin{pmatrix}x_{ij}(\eta)\\y_{ij}(\eta)\\z_{ij}(\eta)\\0\end{pmatrix}, \tag{11.74}$$

where x_{ij}, y_{ij}, and z_{ij} are the x_i, y_i, and z_i displacement components of mode j of link i's deflection, respectively, δ_{ij} is the time-varying amplitude of mode j of link i, and m_i is the number of modes used to describe the deflection of link i

$$\mathbf{E}_i = \left(\mathbf{H}_i + \sum_{j=i}^{m_i}\delta_{ij}\mathbf{M}_{ij}\right), \tag{11.75}$$

where

$$\mathbf{H}_i = \begin{pmatrix}1&0&0&l_i\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{pmatrix} \tag{11.76}$$

and

$$\mathbf{M}_{ij} = \begin{pmatrix}0&-\theta_{zij}&\theta_{yij}&x_{ij}\\\theta_{zij}&0&-\theta_{xij}&y_{ij}\\-\theta_{yij}&\theta_{xij}&0&z_{ij}\\0&0&0&0\end{pmatrix}. \tag{11.77}$$

This description is used in the formation of the potential and kinetic energy contributions of each element of mass. The integral over the link accumulates the total energy for that link and the sum over all links gives the energy of the arm system. Interchanging the order of integration and summation allows key invariant parameters to be identified, such as the *modal mass* that multiplies the second derivative of modal amplitude and the *modal stiffness* that multiplies the amplitude itself. The intermediate steps are aimed at the evaluation of Lagrange's equations, which separate into derivatives related to the rigid variables and the flexible variables. If the rigid variables $\boldsymbol{\theta}$ are the joint variables, two groups of equations result

$$1.\quad \frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial \mathcal{T}}{\partial \dot{\theta}_j}\right) - \frac{\partial \mathcal{T}}{\partial \theta_j} + \frac{\partial \mathcal{V}_\mathrm{e}}{\partial \theta_j} + \frac{\partial \mathcal{V}_\mathrm{g}}{\partial \theta_j} = F_j \tag{11.78}$$

$$2.\quad \frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial \mathcal{T}}{\partial \dot{\delta}_{jf}}\right) - \frac{\partial \mathcal{T}}{\partial \delta_{jf}} + \frac{\partial \mathcal{V}_\mathrm{e}}{\partial \delta_{jf}} + \frac{\partial \mathcal{V}_\mathrm{g}}{\partial \delta_{jf}} = 0 \,, \tag{11.79}$$

where the subscript e indicates elasticity, subscript g indicates gravity, and $\boldsymbol{\delta}$ indicates the amplitude of a mode shape.

The simulation form of the equations is achieved by grouping all second derivatives of the rigid and flexible coordinates, which yields

$$\begin{pmatrix}\mathbf{M}_{rr}(\boldsymbol{q}) & \mathbf{M}_{rf}(\boldsymbol{q})\\ \mathbf{M}_{fr}(\boldsymbol{q}) & \mathbf{M}_{ff}\end{pmatrix}\begin{pmatrix}\ddot{\boldsymbol{\theta}}\\ \ddot{\boldsymbol{\delta}}\end{pmatrix} = -\begin{pmatrix}0&0\\0&\boldsymbol{K}_\mathrm{s}\end{pmatrix}\begin{pmatrix}\boldsymbol{\theta}\\\boldsymbol{\delta}\end{pmatrix} + \boldsymbol{N}(\boldsymbol{q},\dot{\boldsymbol{q}}) + \boldsymbol{G}(\boldsymbol{q}) + \boldsymbol{R}(\boldsymbol{q},\dot{\boldsymbol{q}},\boldsymbol{Q}) \,, \tag{11.80}$$

where $\boldsymbol{\theta}$ is a vector of rigid coordinates, usually the joint variables, $\boldsymbol{\delta}$ is the vector of flexible coordinates, $\boldsymbol{q} = [\boldsymbol{\theta}^\mathrm{T}\ \boldsymbol{\delta}^\mathrm{T}]^\mathrm{T}$, $\mathbf{M}_{ij}$ is the *mass* matrix for rigid and flexible coordinates corresponding to the rigid $(i,j=r)$ or flexible $(i,j=f)$ coordinates and equations, $\boldsymbol{N}(\boldsymbol{q},\dot{\boldsymbol{q}})$ contains the nonlinear Coriolis and centrifugal terms, $\boldsymbol{G}(\boldsymbol{q})$ captures gravity effects, $\boldsymbol{Q}$ represents the externally applied forces, and $\boldsymbol{R}$ captures the effect of external forces and all other nonconservative forces including friction.

On the other hand, the inverse dynamics form of the equations is (assuming negligible friction)

$$\boldsymbol{Q} = \begin{pmatrix}\mathbf{M}_{rr}(\boldsymbol{q}) & \mathbf{M}_{rf}(\boldsymbol{q})\\ \mathbf{M}_{fr}(\boldsymbol{q}) & \mathbf{M}_{ff}\end{pmatrix}\begin{pmatrix}\ddot{\boldsymbol{\theta}}\\ \ddot{\boldsymbol{\delta}}\end{pmatrix} + \begin{pmatrix}0&0\\0&\boldsymbol{K}_\mathrm{s}\end{pmatrix}\begin{pmatrix}\boldsymbol{\theta}\\\boldsymbol{\delta}\end{pmatrix} - \boldsymbol{N}(\boldsymbol{q},\dot{\boldsymbol{q}}) - \boldsymbol{G}(\boldsymbol{q}) \,. \tag{11.81}$$

In general, $\boldsymbol{Q}$ is dependent on the vector $\boldsymbol{T}$ of motor torques or forces at the joints, and the distribution of these effects is such that

$$\boldsymbol{Q} = \begin{pmatrix} \mathbf{B}_r \\ \mathbf{B}_f \end{pmatrix} \boldsymbol{T} \,, \tag{11.82}$$

where $\boldsymbol{T}$ has the same dimensionality as $\boldsymbol{q}_r$. By rearranging the inverse dynamics equation, we obtain

$$\begin{pmatrix} \mathbf{B}_r & -\mathbf{M}_{rf} \\ \mathbf{B}_f & -\mathbf{M}_{ff} \end{pmatrix} \begin{pmatrix} \boldsymbol{T} \\ \ddot{\boldsymbol{\delta}} \end{pmatrix} = \begin{pmatrix} \mathbf{M}_{rr} \\ \mathbf{M}_{rf} \end{pmatrix} \ddot{\boldsymbol{\theta}} + N(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \boldsymbol{G}(\boldsymbol{q}) \,, \tag{11.83}$$

which is arranged with a vector of unknowns on the left-hand side of the equation and the prespecified rigid and solvable flexible coordinates on the right. This is at least approximately true, with the caveat that the nonlinear and gravity terms have a weak influence on the immediate flexible coordinates, which are not simultaneously known but which can be estimated for the next time step.

One consequence of the distributed flexibility is non-minimum-phase behavior. This can occur for other cases where the excitation is separated from the output point by mass and flexibility, a condition known as noncollocation. These systems are characterized by a step response in which the initial movement is opposite in sign from the final movement. This is simply characterized for linear systems as a transfer function zero in the right half-plane. As will be shown in the section on Command Generation, it may be possible to overcome this problem by redefining the output.

11.2.3 Control

Appropriate reaction to flexible dynamics requires appropriate sensors. While a Kalman filter can theoretically observe flexible link states from joint measurements, it is based on back driving the joint from the various modes of flexible dynamics. More reliable indications of flexibility include strain gauges, accelerometers, and optical sensors.

Sensors for Flexibility Control

The strain of a beam element is a direct indication of curvature, which is the second spatial derivative of beam deflection, as indicated in (11.57). Modern semiconductor strain gauges have high gauge factors producing a good signal-to-noise ratio. The second spatial derivative relationship means that a slight change in location can result in a large change in the reading, but once affixed to the arm this is not a problem. More of a problem is the relatively short lifetime of a strain gauge subject to wear and tear if not carefully protected. By suitably placing strain gauges, good measurements of multiple assumed modes are possible.

Optical measurement of deflection can be very versatile but is subject to interference between a target measurement point and the sensor, particularly if the sensor is stationary. If the sensor is mounted on the link, this difficulty is reduced but optical measurements are then usually a complex combination of the rotation of the optical axis of the sensor and the deflection of the target. Machine vision senses a wide field of view but typically with less precision. Multiple points can be detected, which is attractive in measuring multiple modes. Machine vision requires considerable processing and, even for enhanced targets or retroreflective fiducials, sample rates can be slow relative to some modes of control interest.

Accelerometers are attractive transducers in that they do not require explicit grounding to a fixed frame. Microelectromechanical systems (MEMSs) accelerometers provide the required low-frequency sensitivity and are very inexpensive. However, they are subject to orientation in a gravity field, which must be compensated for by knowing the orientation of the sensor. If one is interested in position, accelerometers require two integrations and are consequently subject to drift in position measurement. Consequently, the combined use of sensors is appealing. For example, vision sensors, which are relatively slow but give direct position information for the end of a flexible member, can be combined with accelerometers, which give information about the same point at high sample rates but with noise and drift problems. The two sensor readings can be combined with a Kalman filter or other fusion scheme.

Issues Related to Model Order

Realizing that the theoretical order of a flexible arm is infinite, one should be prepared to deal with certain issues related to model order. The first of these is possible aliasing problems arising from finite sampling rates. Antialiasing filters should be employed, and even more effective is the use of analog damping by materials, damping treatments, or the back electromotive force (emf) of actuators to provide an inherent damping of the system under digital control. A more subtle problem is that of measurement and control spill over.

Control of Special Configurations: Macro/Micro Redundancy

Several special configurations have been proposed to enable improved controlled performance of flexible link arms. The problem with noncollocation noted above is a case in point. In some cases, a large range of motion and high accuracy cannot be achieved with a single set of actuators. Redundant degrees of freedom can then

be employed with the macro degrees of freedom that provide large range of motion but with inherent lack of precision or accuracy. Vibrations are particularly difficult to eliminate with a long arm, and construction accuracy and even thermal drift can exceed the tolerance limits. If the gross motion is completed quickly, residual vibrations may remain and be very lightly damped. Actuators constructed for gross motions of large axes do not have the bandwidth capabilities to damp these vibrations actively, but small motion axes typically do. This has been illustrated in several instances for long-reach manipulators when exploring nuclear waste cleanup solutions [11.71–74]. Using the inertial forces generated at the end of the arm also has the advantage of creating forces collocated with the measurements being used for control, assuming the base of the micro arm is at the end of the macro arm and the location of sensors. *Book* and *Loper* [11.73] and *George* and *Book* [11.75] have shown that this control is possible based on accelerometers in multiple degrees of freedom. It is also possible to excite undesired modes which have not been accounted for. If the full six degrees of freedom are used it is possible to argue stability based on a simple passive analogy. Generating prescribed forces at the base of an articulated micromanipulator is a challenging task, requiring the solution of a new type of inverse dynamics problem, which solves for motions or joint torques to produce the prescribed base forces and moments.

Command Generation for Flexible Link Arms via Command Shaping

The trajectory undertaken by a flexible arm can dramatically affect the consequences of flexibility. Purely open-loop anticipation of the flexible dynamics has been used to create motion profiles wherein the motion cancels the incipient oscillation that had been created by earlier motions. This strategy is referred to as command shaping. A more complicated strategy to implement is a true inverse-dynamics-based trajectory wherein the rigid and flexible states are all orchestrated to arrive at the desired point in state space, typically at a static equilibrium. The high order of the flexible model is further complicated by the non-minimum-phase nature of the noncollocated system and noncausal inverse is necessary to produce a physically realizable trajectory (VIDEO 778).

Time delay command shaping is a variation of the deadbeat control introduced by [11.76, 77]. It can be couched in terms of a finite impulse response filter with a simple impulse response as depicted in Fig. 11.11 with a single input impulse producing a small number (typically two to four) of output impulses with appropriate time spacing and amplitudes. Extensions by *Singer* and *Seering* [11.78] and *Singhose* et al. [11.79] have produced more robust varieties under the trademark Input Shaping. The robustness is enhanced by using more output impulses and appropriate selection of the impulse spacing. The optimal arbitrary time-delay (OAT) filter [11.80, 81] reduces this selection to a simple formula for linear vibrations that depends on the natural frequency and damping ratio of those modes to be eliminated from the response. The simplicity of implementation is illustrated in Fig. 11.12, where the relationships between the filter parameters and the mode parameters are

$$\begin{aligned}
\text{coefficient 1} &= \frac{1}{M} \\
\text{coefficient 2} &= \frac{-(2\cos(\omega_\mathrm{d} T_\mathrm{d})\,\mathrm{e}^{-\zeta\omega_\mathrm{n} T_\mathrm{d}})}{M} \\
\text{coefficient 3} &= \frac{\mathrm{e}^{-2\zeta\omega_\mathrm{n} T_\mathrm{d}}}{M} \\
M &= 1 - 2\cos(\omega_\mathrm{d} T_\mathrm{d})\,\mathrm{e}^{-\zeta\omega_\mathrm{n} T_\mathrm{d}} + \mathrm{e}^{-2\zeta\omega_\mathrm{n} T_\mathrm{d}} \\
\omega_\mathrm{d} = \omega_\mathrm{n}\sqrt{1-\zeta^2} &= \text{damped natural frequency} \\
\omega_\mathrm{n} &= \text{undamped natural frequency} \\
\zeta &= \text{damping ratio} \\
T_\mathrm{d} &= \text{time delay selected, an integer number of samples .}
\end{aligned} \tag{11.84}$$

Note that the formulation here is oriented toward digital systems with fixed sampling intervals such that the time delay of the filter is adjusted to an integer

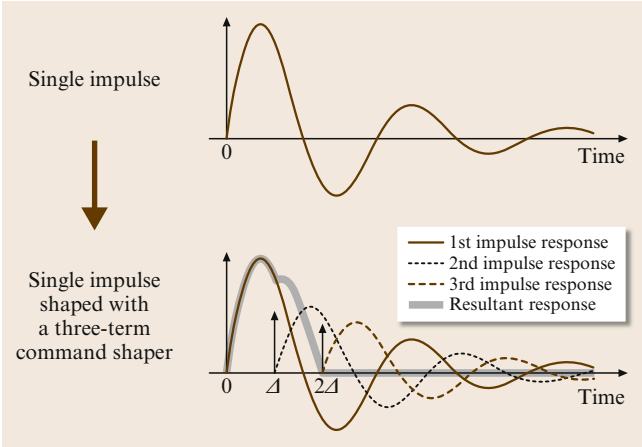


Fig. 11.11 Impulse response of a shaper and the effect on a vibratory system. Time is normalized by the intersample time T_s and T_d, chosen such that $\Delta = T_\mathrm{d}/T_\mathrm{s}$ is an integer

Avidbots Ex1016
Page 314 of 2154

number of sampling intervals. Command shaping has been applied to a wide range of flexible systems including disk drives, cranes, and large robot arms, such as the example by CAMotion (VIDEO 777) as shown in Fig. 11.13. Note that this example contains a flexible drive belt, effectively a flexible joint, as well as flexible links that contribute to the compliance of the lowest mode that is quenched with an OAT filter.

Adaptive versions of command shaping have also been demonstrated. A version oriented toward repetitive motion applications is presented in [11.82].

Command Generation for Flexible Link Arms via Inverse Dynamics

In contrast to the scant system knowledge required for command shaping, the inverse dynamics of a flexible arm requires a great deal of knowledge about the system. The creation of an appropriate model was discussed above. An assumed modes model, for example, is well adapted to inverse dynamics if the rigid coordinates are not the joint angles, but the angles between lines connecting the joint axes (where it is assumed that joints are rotational). In this case, the desired end-of-arm motion can be expressed solely in terms of rigid variables, that is, the same variables $\boldsymbol{q}_{\mathrm{d}}$ used in the flexible joint case, and the desired flexible variables are found from the second-order equations driven by the known rigid positions and velocities. The remaining difficulty is the unstable zero dynamics of the system.

For the purposes of introduction, only the linear situation will be treated here in the fashion of [11.83]. First, the rigid coordinates and their derivatives are found from the desired end-of-arm motion profile. The flexible equations are separated into causal stable zeros and anticausal unstable zeros, and the stable inverse is solved forward in time. Physically the motion response responds after the torque inputs. The computational input however is the response and the torque is calculated as an output, so no physical laws are violated in solving the anticausal portion beginning at the final time when flexible coordinates are specified to be zero and proceeding to the initial time and beyond. In fact, the exact solution extends from negative infinity (where the anticausal solution decays) to positive infinity (where the causal solution decays). For realistic dynamics, the necessary time for the input is only slightly more than the motion time for the end of the arm because the zeros are often quite far from the origin in both directions. If the end-of-arm motion begins at time zero, shortly before that, the joint torques begin to *pre-shape* the flexible links without moving the end of arm. Similarly, when the end of the arm reaches its final position, the joint torques must release the arm deflection in a manner that is not seen from the end of arm (or any other chosen output location). Figure 11.14 shows the example from *Kwon* [11.84] for a single-link arm and the VIDEO 778 shows the behavior improvement.

Since inversion of the flexible plant as described earlier containing nonminimum phase or *unstable* zeros can result in an anticausal input torque to achieve an arbitrary trajectory, it is appropriate to look for other trajectories with special characteristics that avoid these

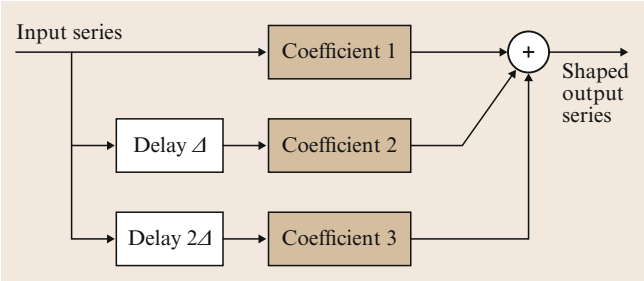


Fig. 11.12 Block diagram representation of the command shaping algorithm

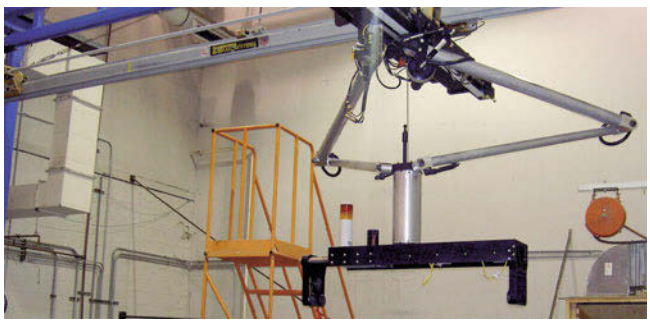


Fig. 11.13 A large flexible linking arm with command shaping

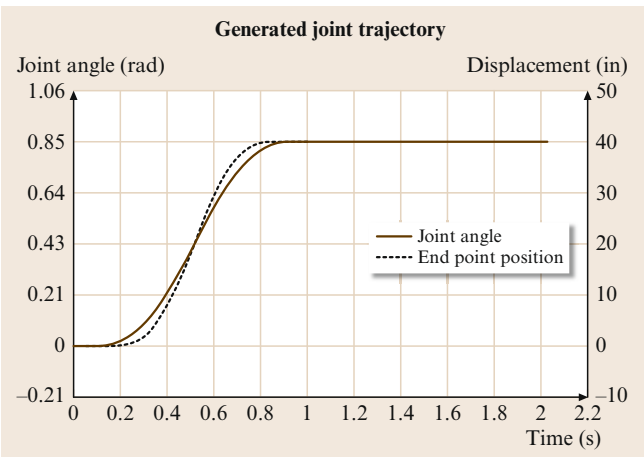


Fig. 11.14 Inverse dynamics driven motion of a single-link flexible arm

problems. The location of zeros are dependent on where on the flexible link the trajectory is specified. *De Luca* et al. studied the one flexible link rotational arm and the migration of zeros as the point of interest is shifted from tip to joint. They utilize a point where zeros are nonexistent as the output point, a point that enables inversion to proceed without the causality problem [11.85–87]. When this point reaches the desired angle with zero deflection, the end point will also be at the desired angle. Joint velocity and deflection velocities must also be brought to zero resulting in a rest-to-rest motion completed in specified time and with no residual oscillations. The cited references show the equivalence of this development in the time or frequency domain for linear systems. For a two link arm (only the forearm is flexible), the nonlinear behavior demanded a time domain approach [11.86].

De Luca and *Di Giovanni* [11.85] address rest-to-rest motion of a single flexible link and carries through an example for a single flexible mode. The later reference [11.87] also provides experimental results for which a brief video (VIDEO 779) is available. The strength of this approach is its relative simplicity in achieving the desired final position described by a single polynomial. The degree of the single polynomial required to meet all the boundary conditions is $4n_e + 3$ where n_e is the number of flexible modes. One difficulty is the inability to efficiently constrain the peak torque required. Torque profiles developed in this simplest of strategies have high peaks but for most of the trajectory (at start and end) are relatively low. Hence the peak torque capabilities of the actuator may not be efficiently used. It is not necessary for a single polynomial to describe the entire trajectory, of course, but the complexity of developing the response without zeros becomes more challenging for piece wise polynomials, and arises from meeting all the boundary conditions on polynomial segments. The nonlinear case arising from a two link arm with the forearm flexible [11.86] provides further exploration of the inversion approach for flexible arms.

The model employed by De Luca varies slightly since deflection is measured from an axis passing through the dynamic center of mass of the flexible arm as shown in Fig. 11.15, not the tangent to the neutral axis of the beam at the rotary actuator. This modifies the boundary conditions on the shape equation and hence the mode shapes. For an introduction, a single flexible link is considered here. Variables are defined as follows:

- J_0, J_p: rotary inertia of the rotary actuator, tip payload, respectively
- m_p: mass of the tip payload
- $\theta, \theta_c, \theta_t$: angles through center of mass, tangent to actuated end, and through tip, respectively
- x: axis through the center of mass (COM)
- y: deflection perpendicular to x.

Consider the case with no damping on the modes. While modal damping is readily incorporated, its typically low values justify an abbreviated treatment here for the sake of space and simplicity. The output y specified is the sum of the angle (to the center of mass), the angular velocity, deflections and deflection rate, each times appropriate gains. These gains are found by specifying the transfer function from joint torque to y to be of maximum relative degree, i. e., it has no zeros. Equivalently, the successive derivatives of y do not explicitly involve the torque until the $(2n_e + 1)$-th derivative. Note that these characteristics ascribed to differentially flat systems may yield a more complete understanding of this approach [11.88, 89],

$$\frac{y(s)}{\tau(s)} = \frac{K}{s^2 \prod_{i=1}^{n_e} \left(s^2 + \omega_i^2\right)} ,$$

$$K = \frac{1}{J} \prod_{i=1}^{n_e} \omega_i^2 .$$

When transformed to the time domain, the resulting form of the torque based on derivatives of the desired output will be

$$\tau_d(t) = \frac{J}{\prod_{i=1}^{n_e} \omega_i^2} \left[y_d^{2(n_e+1)}(t) + \sum_{i=0}^{2n_e-1} \alpha_i y_d^{(i+2)}(t) \right] .$$

The α_i values are readily found from the convolution of the polynomial or expansion of the denominator of. $y(s)/\tau(s)$. But what must be used for y_d? With the given definition of y, the desired value of y_d for the rest-to-rest trajectory will be achieved by creating an interpolating trajectory with the desired initial and final values of y_d

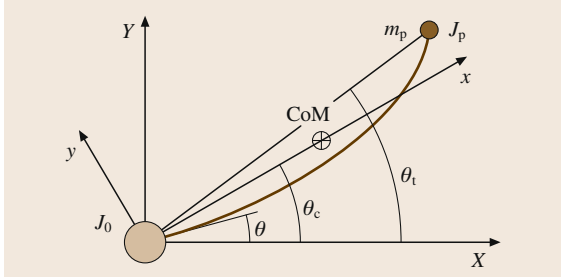


Fig. 11.15 A definition of variables for the causal inversion of De Luca

and derivatives of y_d at the initial and final times that are zero. The number of derivatives to set equal to zero are at least $2n_e + 1$, but it may be preferred to set additional derivatives equal to zero to obtain a smoother torque at the ends of the trajectory. As explained in [11.85] this is a transformation of states that results also in the desired rest-to-rest behavior of the joint angle, the deflection and their derivatives. For more robustness, the feedforward command should be supplemented with feedback control, for example a simple PD or PID control of the joint angle to follow y_d.

Shown in Fig. 11.16 are the torque and resulting bending deflection for a 2.2 s move using this combination on a single link flexible arm [11.87].

To best comprehend these results, some details of the flexible arm in the experiment are needed. The desired trajectory was defined by a polynomial of degree 19. The arm length was 0.655 m, of sheet steel 2 mm thick and 51 mm wide, directly driven by a DC motor with inertia of $1.888 \times 10^{-3}\,\mathrm{kg\,m^2}$ and no tip mass. The resulting natural frequencies were 14.4, 13.2 and 69.3 Hz with damping ratios of 0.0001, 0.001, and 0.008, respectively. Frequency errors of the model were less than 1%. Friction compensation was applied based on static and viscous models of the friction under continuous rotation.

Feedback Control of Flexible Links

Flexibility in the links is in some cases controlled in the same way as concentrated flexibility in joints. Herein the focus is on approaches that apply specifically to link flexibility; consequently the use of Kalman filtering to estimate tip position and other broadly used techniques are not reexamined. A long history of research on this topic exists. *Cannon* and *Schmitz* [11.90], *Truckenbrot* [11.91], and others showed that these techniques work for flexible link arms and that the noncollocated nature of the system aggravates the sensitivity to modeling errors.

Strain measurements provide a valuable indication of the state of a flexible link, and strain rate, obtained from filtered differentiation of the typically noisy strain signal, was shown by *Hastings* and *Book* [11.92] to be an appropriate way of overcoming the limitations imposed by purely joint feedback control. *Yuan* et al. [11.93] produced an adaptive version of strain-rate feedback.

Strain measurements readily lend themselves to calculation of the modal amplitudes, and the number of appropriately placed strain gage measurements N can be equal to the number of mode amplitudes to be evaluated. In this case, no estimation algorithm is necessary, although filtering is usually needed. If the mode shape φ_i of a beam is known, strain is proportional to the second derivative of φ_i

$$\alpha_{ij} = \left.\frac{\partial^2 \varphi_i}{\partial x^2}\right|_{x_j} . \tag{11.85}$$

By inverting the coefficient matrix thus formed, it is simple to obtain the amplitude of N modes as

$$\begin{pmatrix} \delta_1 \\ \delta_2 \\ \vdots \\ \delta_N \end{pmatrix} = \begin{pmatrix} \alpha_{11} & \alpha_{12} & \dots & \alpha_{1N} \\ \alpha_{21} & \alpha_{22} & \dots & \alpha_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_{N1} & \alpha_{N2} & \dots & \alpha_{NN} \end{pmatrix}^{-1} \begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_N \end{pmatrix} . \tag{11.86}$$

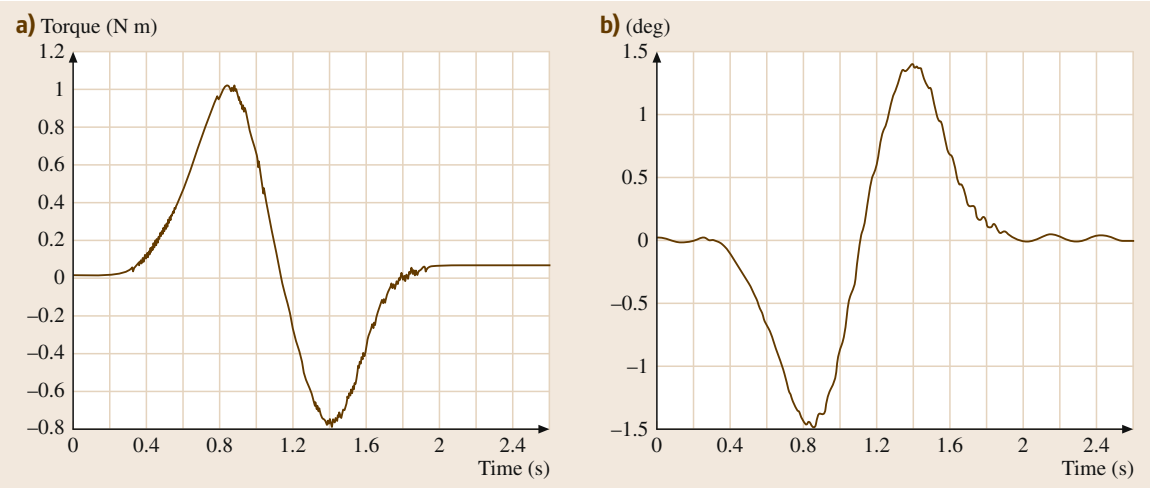


Fig. 11.16 **(a)** Applied torque for rest to rest motion with PD control. **(b)** Bending deformation at the tip

Additional attention is necessary if beam torsion is a significant part of the mode deflection, but the principle still applies. Active control of high-frequency modes may not be warranted, and feedback of the one or two lowest-frequency modes can be managed by classical or state-space design techniques to increase the damping of that mode.

The use of the measurement of the end-of-arm position has perpetual appeal and is very effective for rigid arms for the elimination of variability due to arm distortion and the change over time of arm calibration. Machine vision can eliminate uncertainty in the target position as well as the arm position. These approaches are plagued by the noncollocated and hence non-minimum-phase dynamics of link flexibility, however, and may require particular caution. *Wang* and *Vidyasagar* [11.94] proposed an alternative formulation of EOA measurements in the control formulation that he called the reflected tip position. For a rotating link of angle θ and length l, the deflected position is $l\theta + \delta$ while the reflected tip position $l\theta - \delta$ is minimum phase. This combination of joint and tip measurements produced the desired undeflected position but would not serve for EOA tracking control. *Obergfell* and *Book* [11.95] synthesized an output based on strain feedback combined with EOA position that allowed tracking control as well. They first produced a passive system with deflection feedback obtained from optical measurements of the link deflection, and then devised a traditional feedback control of the tip position measured by machine vision of the large two link arm shown in Fig. 11.17 for the experimental robot RALF (robot arm long and flexible). Figures 11.18 and 11.19 compare the path for a large square traced in a plane with only the deflection feedback and with both deflection and EOA position feedback.



Fig. 11.17 Robotic arm large and flexible (RALF) used in EOA feedback experiments

Robust Estimation for Flexible Link Systems

The state feedback algorithms readily designed for linear flexible link systems are dependent on information about states that cannot be directly measured and are theoretically infinite in number. Truncation of the states to represent the most dominant low frequency modes is a ready solution to the second problem and state estimators or observers present a viable approach to the first problem. (Estimator and observer are commonly used interchangeably, but the term observer is more common will be used in this section.) For linear systems a Luenberger observer or the stochastically based Kalman filter are commonly used. Unfortunately, observers require an accurate system model or the estimates may be poor and the observer, a dynamic system

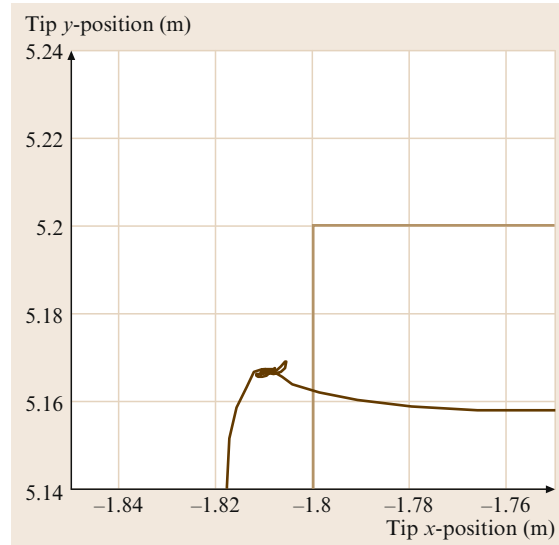


Fig. 11.18 Path followed without EOA position feedback. (Commanded trajectory is a *square* partially shown in *light brown*. Actual trajectory shown in *dark brown*)

of order as high as the order of the system itself, may even be unstable. Hence the robustness of the state observer is a critical subject for the control of flexible link systems. The quality of the control is ultimately the best measure of the quality of the estimator. Robust observer design has been the subject of considerable research and, in large part, mirrors the development of robust controllers. This is natural since an observer is a dynamic system which, like the plant itself, can be controlled to achieve a desired outcome. *Control effort* of the observer is utilized to drive convergence of the estimated outputs of the observer to the measured outputs of the real system. Hence research on estimators based on H-infinity, fuzzy logic, neural network, sliding mode, adaptive, and other design techniques abound. In this chapter, two types of observers will be discussed: sliding mode observers and multiple model observers.

For all observers the placement of sensors is crucial and for estimation of multiple modes it is desirable, although not theoretically essential, to use multiple sensors. Joint sensors (position and velocity) are typically available for general pose (joint) control, but for deflection sensing strain gages and accelerometers are currently most appropriate. The MEMS accelerometers now readily available and inexpensive will be the principle choice here. Strain gages are more fragile and generally provide more complex readings of the mode behavior. Accelerometers are complicated by the fact that acceleration is not dependent on just states z which include the mode amplitudes and their derivatives, but also on the input u creating the link motion. Acceleration measurement of a point p on the link is affected by both the states z and the input u as follows,

$$\ddot{x}_p = \left\{-\beta(p)[\Phi][\omega^2] - \beta(p)[\Phi][\hat{C}]\right\} z + \left\{\beta(p)[\Phi][\Phi]^T Q\right\} u \,,$$

where $\ddot{x}_p$ is the acceleration in the direction perpendicular to the beam axis, Φ is the matrix of eigenvectors, z is the state vector, composed of joint motion as well as beam deflections, p is the location of the sensor along the beam axis, ω^2 is the diagonal matrix of natural frequencies squared, $\hat{C}$ is the modal damping matrix, $\beta(p)$ expresses the sensor motion in terms of the modal states variables, and Q relates the input to the system states.

As *Post* [11.96] has explained, the ability of a given sensor to detect a mode depends on the amplitude of the mode at the sensor location. Overall the singular values of the observability grammian, also used for testing the system observability, is a reasonable but not perfect metric. A robust measurement requires the placement of sensors so as to avoid nodes (zero crossings) of the modes to be sensed even when parameters of the system change or are inaccurate. A single sensor (strain gage or accelerometer) is much more likely to have this problem than multiple sensors. The next sections depend on suitably placed sensors. For those modes not being controlled, one should minimize their influence by placing sensors near the node of those modes. This helps to avoid spillover of these modes that interferes with the system as approximated.

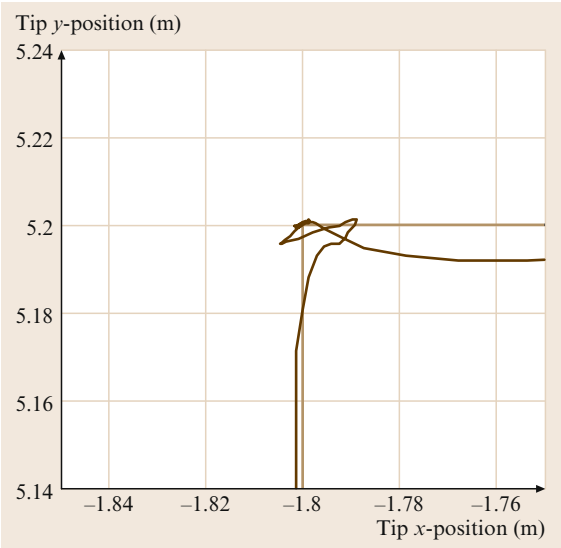


Fig. 11.19 Path followed with EOA feedback from camera. (Commanded trajectory is a *square* partially shown in *light brown*. Actual trajectory shown in *dark brown*)

Parametric Errors in System Model

Parametric errors are inaccuracies in the parameters of a model with an accurate representation of the system structure. For flexible link systems this might arise from inaccurate payload mass, stiffness values, stiffness distribution, boundary conditions, etc., all of which lead to inaccurate mode shapes, natural frequencies and input coefficients. Errors are inevitable to some degree. As a consequence of parametric errors, one of the most powerful results of linear feedback control of estimated states, the separation principle which says the design of the observer does not change the eigenvalues of the controlled system and vice versa, is compromised. The rapid convergence of observer error to zero will in general not occur with parametric error. The net effect is a major blow to the effectiveness of the classical Luenberger observer. The Kalman filter, which optimizes observer gains under conditions of Gaussian process and measurement noise, is likewise

compromised. Without good state estimates the effectiveness of the powerful state feedback approaches are in jeopardy.

One currently popular tack to improve the robustness of observers is the sliding mode observer. There are several variations of this approach, and the technique by *Wallcot* and *Zak* [11.97, 98] was followed by *Post* [11.96] in exploring the consequences of parametric error. As typical, a sliding surface is defined which is intended to guide the estimate error to zero. The reaching phase moves from the initial error state to the sliding surface and when that surface is crossed the sign on the control changes. This rapid switching may be modified by establishing a *boundary layer* near the switching surface that substitutes a linear behavior.

In this section, we will use z as the state vector, $\hat{z}$ as its estimate, L as the feedback of the observer, y as the measurement and K_S as the sliding mode gain. The *hat* on a variable will generally denote its estimate and an over dot its time derivative. The symbol Δ preceeding a symbol indicates the variation of the true system parameter from its model. The linear plant model is thus

$$\dot{z} = Az + Bu\,; \quad y = Cz + Du\,.$$

The equation approaching the switching surface is

$$\dot{\hat{z}} = A\hat{z} + Bu + L(y - \hat{y}) + K_s \mathrm{sgn}(y - \hat{y})\,.$$

When the boundary layer is included, the control is changed in a region near the switching surface where the design is essentially a conventional linear observer. For present purposes we will not employ a boundary layer but examine the pure sliding mode case, for which the error equation is

$$\dot{e} = A_0 e - K_s \mathrm{sgn}\,(Ce) + \Delta Az + \Delta Bu\,,$$

which clearly has a steady state error solution that is not zero except in the case that $\Delta Az = -\Delta Bu$ which tends to occur when the overall system comes to rest. Thus the convergence of the sliding mode observer has similar issues found with the conventional linear observers. They can be stable but can converge to the wrong steady state value leading to error, particularly after the observer's state estimates should have converged to the slower changing system state. However, when the overall plant reaches steady state and $z = 0$, $u = 0$, the observer error will also converge to zero. In contrast, the usual intent is to design the observer to quickly converge to the true state estimate in order to appropriately control the plant.

Successful Estimation with Model Parameter Uncertainty

Models are seldom, if ever, exact in their representation of a physical system. In addition to error in parameter identification, parameters can change with time and operating conditions. For example, robots acquire and release payloads and move to different poses which may change parameters. As reported before, convergence of state observers is adversely affected by these changes which can lead to poor performance and even instability. In a recent work, *Post* [11.96] described a multiple model estimation scheme that successfully overcomes these limitations for a flexible link robot in both simulation and experiment. Other authors, e.g., *Chalhoub* and *Kfoury* [11.99] have used fuzzy logic techniques to address the needs for improving sliding mode observers.

Since the true state of the system is not known, the steady state estimation error cannot be calculated, but if it is known to exist, Post has shown it to be proportional to parameter error. Hence multiple models based on a different set of sensors can be used to establish which is most accurate. The difference in estimated states based on two sensor selections $\delta_z(t) = \hat{z}_{s_1} - \hat{z}_{s_2}$ is then a metric for choosing between alternative observers in an array of observers with different assumed parameters.

The observers themselves (conventional Luenberger observers or Kalman filters) have the form

$$\dot{\hat{z}}_{s_1}(t) = A\hat{z}_{s_1}(t) + Bu(t) + L_{s_1}\left[y_{s_1}(t) - \hat{y}_{s_1}(t)\right],$$

where $\hat{z}_{s_1}$ is the estimate of the state vector z using the sensor array 1, A is the plant matrix, B is the input matrix, L_{s_1} is the gain matrix to be designed, y_{s_1} is the sensor measurement of sensor array 1, and $\hat{y}_{s_1}$ estimate of sensor measurement of sensor array 1, and similarly for sensor s_2. The difference between states and their estimates converges rapidly (relative to the states themselves) to steady state and hence $\dot{\hat{z}}_{s_1} - \dot{\hat{z}}_{s_2}$ converges to zero and hence

$$\begin{aligned}\delta_z(t) = {} & \chi_1[\Delta Az(t) + \Delta Bu] \\ & + \chi_2[\Delta C_{s_1} z(t) + \Delta D_{s_1} u(t)] \\ & + \chi_3[\Delta C_{s_2} z(t) + \Delta D_{s_2} u(t)]\end{aligned}$$

is found to be the *steady state* slowly changing (relative to the observer dynamics) difference in the two observers based on sensors s_1 and s_2. Here, the χ_i are constant matrices based on the model parameters and not the true values of the system. Δ indicates the deviation of the true matrices from the model's matrices following that symbol. Note that subscripts s_1 and s_2 indicate different sensors involved in the measurement

and feed through matrices C and D, respectively. Also note that this equation is somewhat complicated by the feedthrough matrices occurring due to the use of accelerometers to sense the motion.

Incorporating the specific representation of a linear flexible manipulator, the equations are

$$\dot{z} = \begin{pmatrix} 0_{(n\times n)} & I_{(n\times n)} \\ -\omega^2_{(n\times n)} & -\hat{C}_{(n\times n)} \end{pmatrix} z + \begin{pmatrix} 0_{(n\times m)} \\ \hat{Q}_{(n\times m)} \end{pmatrix} u \,,$$

where $\hat{C}$ denotes modal damping and $\hat{Q}$ denotes the modal forcing coefficient. The sensor's output is

$$\begin{aligned} y_{Si} = \Big\{ &- [1, \psi_1(p_{si}), \ldots, \psi_n(p_{si})]\, \Phi_{(n\times n)} \omega^2_{(n\times n)} \\ &- [1, \psi_1(p_{si}), \ldots, \psi_n(p_{si})]\, \Phi_{(n\times n)} \hat{C}_{(n\times n)} \Big\} z \\ &+ [1, \psi_1(p_{si}), \ldots, \psi_n(p_{si})]\, \Phi_{(n\times n)} \hat{Q}_{(n\times n)} u \,. \end{aligned}$$

Let $\beta_{Si} = [1, \psi_1(p_{si}), \ldots, \psi_n(p_{si})]$ and one can attribute to the sub-matrices the parametric change matrices

$$\begin{aligned} &(\Delta A z(t) + \Delta B u) = \\ &\quad \begin{pmatrix} 0_{(n\times n)} \\ I_{(n\times n)} \end{pmatrix} \left[\left(-\Delta\omega^2 - \Delta\hat{C} \right) z(t) + \Delta\hat{Q} u(t) \right], \\ &(\Delta C_{Si} z(t) + \Delta D_{Si} u) = \\ &\quad \beta_{Si} \Phi \left[\left(-\Delta\omega^2 - \Delta\hat{C} \right) z(t) + \Delta\hat{Q} u(t) \right], \\ &\delta_z(t) = \left[\chi_1 \begin{pmatrix} 0_{(n\times n)} \\ I_{(n\times n)} \end{pmatrix} + \chi_2 \beta_{s_1} \Phi + \chi_3 \beta_{s_2} \Phi \right] \\ &\quad \times \left[\left(-\Delta\omega^2 - \Delta\hat{C} \right) z(t) + \Delta\hat{Q} u \right], \end{aligned}$$

which will be coalesced to give

$$\delta_z(t) = \Lambda_{(2n\times n)} \gamma(t) \,,$$

where it can be seen that Λ consists of known model parameters and γ is an unknown metric of parameter errors that can't be solved from this equation. However, using the pseudo inverse we can produce a least squares metric $\gamma_{\mathrm{m}}(t)$ which evaluates the relative accuracy of two observers based on different models

$$(\Lambda^{\mathrm{T}}\Lambda)^{-1} \Lambda^{\mathrm{T}} \delta^*_z(t) = \gamma_{\mathrm{m}}(t) \,.$$

The smaller the norm of $\gamma_{\mathrm{m}}(t)$, the more accurate the observer based on a given model is expected to be.

While various methods of using $\gamma_{\mathrm{m}}(t)$ can be contrived, *Post* [11.96] utilized a switching approach he referred to as the multiple model switching adaptive estimator (MMSAE), visualized in Fig. 11.20. The weighting value w_i is based on $\gamma_{\mathrm{m}i}(t)$ for the observer

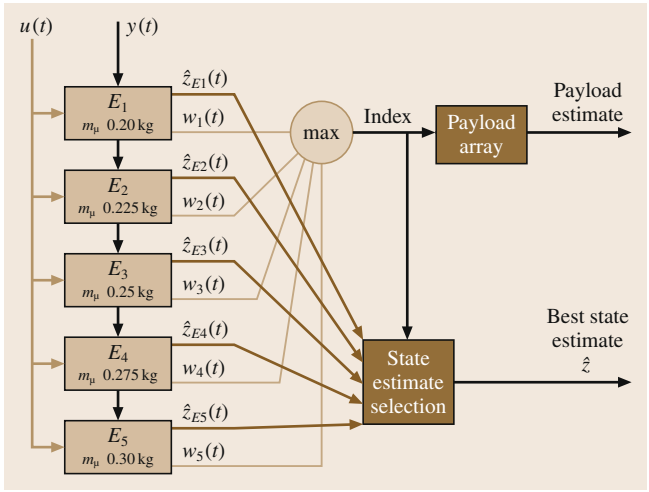


Fig. 11.20 Flow chart for the multiple model switching adaptive estimator (MMSAE)

based on a given model, then the estimate $\hat{z}_i$ from that observer was used. Figure 11.21 shows schematically the flow inside the estimator blocks E_1 through E_5. The parameter varied in Post's experiments was arm payload. Multiple parameters could be varied but would lead to a higher dimensional formulation and poten-

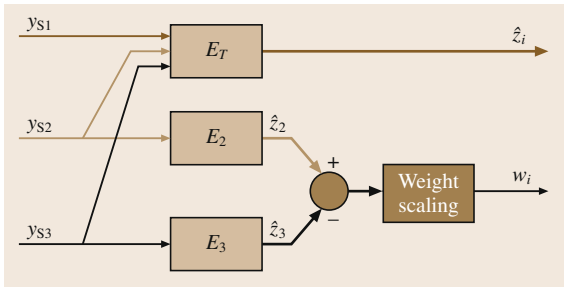


Fig. 11.21 Estimation module with weight scaling

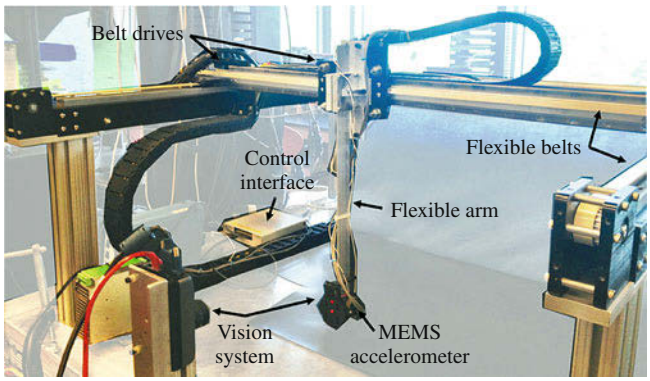


Fig. 11.22 Gantry robot used in MMSAE experiments (after *Post* [11.96])

tially many estimators numbering $N_T = N_1 N_2 \cdots N_p$ for the variation of p parameters, each with N_i values represented.

Post [11.96] conducted experiments with a linear model of the gantry robot as illustrated in Fig. 11.22 and schematically modeled in Fig. 11.23, with the parameters given in Table 11.1 and sensor placements at 0.015, 0.25, and 0.42 m from the actuated end of the beam. He compared:

- Conventional PID (VIDEO 780).
- Fixed gain linear-quadratic regulator (LQR) plus Luenberger state space designs (VIDEO 781).
- MMSAE design observers and LQR controls. Five potential payloads were chosen with spacing in even increments between 0.2 and 0.3 kg, i. e., an endpoint mass between 0.35 and 0.55.

The penalty matrices were kept the same for the LQR controls but the resulting gains changed based on the model employed by the MMSAE selection. While various measures of performance could be demonstrated involving somewhat arbitrary selection of controller types, controller gains and metrics, the following characteristics are considered most critical:

- Stability with various payloads.
- Cycle completion time, with requirements for settling of vibration amplitude at prescribed stops.
- Variation of cycle times with changing payloads.

Since reasonable work cycles often pick up and drop off payloads, the test included these operations. Cycle times are graphed in Fig. 11.24. The light bars show the cycle times for the complete cycle with segments of the task carrying the payload and others with the gripper empty. The dark bars show operation with no payload carried and only the gripper mass present for the entire cycle. In addition to the pair of bars labeled *PID* and *MMSAE* (with obvious meanings) are pairs labeled *SS 0.25* through *SS 0.40*. These latter labels refer to the *designed-for* tip mass (payload plus gripper). The most obvious message from this figure is that the state space control is a vast improvement over PID, reducing cycle time by about 90%. What is not obvious from this figure is that some means of gain change is needed to accommodate the payload variations. With a designed-for tip mass of 0.45 kg and above, the operation without payload was unstable. This is still below the actual tip mass of 0.5 kg. Hence MMSAE was essential for a successful completion of the task with these design criteria (i. e., LQR penalty matrices and resulting gains). The exact tip mass does not need to be known, only the range of masses. The near identical cycle times with and without the additional payload seen on the chart for MMSAE is also a practical advantage.

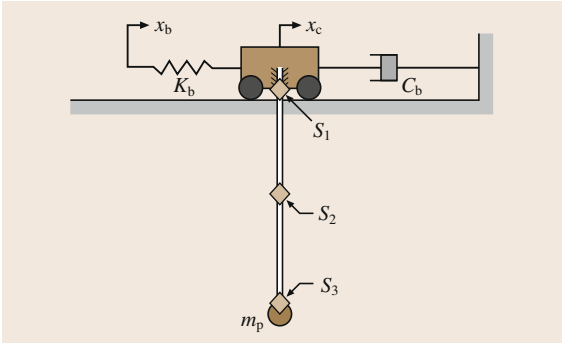


Fig. 11.23 Schematic of gantry robot model

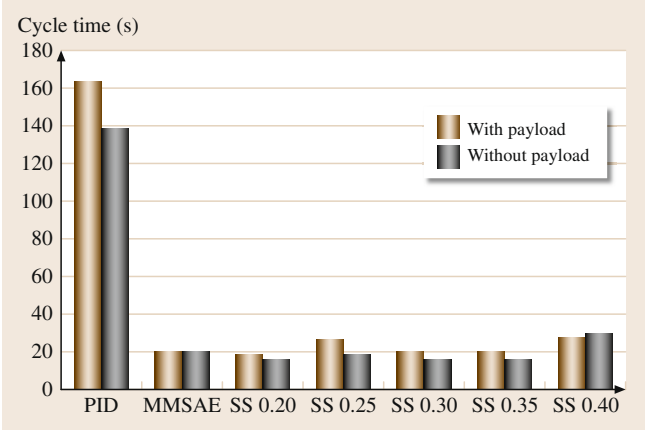


Fig. 11.24 Cycle times with implemented control methods

11.2.4 Further Reading

The topic of flexible link robot control is widely covered with papers on modeling and control. The reader wishing to probe further may want to begin with a paper giving some perspective on the nature of the problem. This was the goal of *Book* in [11.100]. Progress to date in this field from the theoretical perspective is signifi-

Table 11.1 Parameters of the gantry robot model

Parameter	Value	Unit
Payload (m_t)	0.281	kg
Cart mass (m_c)	10	kg
Arm length (L)	0.42	m
Elastic modulus (E)	7×10^{-10}	N/m
Density (ρ)	2700	kg/m^3
Area moment (I)	1.0114×10^{-10}	m^4
Belt stiffness (K_b)	2.1814×10^5	N/m
Cart damper (C_d)	100	N s/m
Structural damping coefficient (γ)	0.0005	N s/m

cant and has been collected by *Canudas de Witt* et al. in [11.101]; this reference includes a wide range of research approaches, but the examples of the practical application of this theory are more limited. The German Aerospace Center has produced lightweight space arms [11.102] and deployed them in space. The startup CAMotion, Inc. [11.103] has produced large lightweight industrial robots using the command-shaping approach to improve cycle time. High-speed arms such as the FlexPicker [11.104] achieve a light weight and avoid flexibility because of their parallel structure, but suffer the usual disadvantages of parallel actuation: restricted workspace considering the space occupied by the robot structure.

Trautt and *Bayo*'s early exploration of non-minimum-phase inverses is described in [11.105]. This is a particularly interesting variation of the work by *Kwon* [11.83, 84] in that it uses the frequency domain and deals with nonzero initial conditions. The concept of using two time scales, a fast time scale for the flexible dynamics and a slow time scale for the rigid dynamics, can be found in *Siciliano* and *Book* [11.106]. The singular perturbation approach was used earlier by Spong for elastic joint arms. *Ghorbel* and *Spong* also consider flexible links and contact with a rigid environment in [11.107]. Achieving robustness is a problem, particularly with end-point sensors. Sliding mode control [11.108] and passivity-based control [11.109] address this problem in effective but widely varying ways. While these seem to be the more promising methods of control, one will find many more approaches to this challenging control problem. H_∞, adaptive, fuzzy, and other controllers have all been applied with varying degrees of success.

Video-References

VIDEO 133 Cartesian impedance control with damping off
available from http://handbookofrobotics.org/view-chapter/11/videodetails/133

VIDEO 134 Cartesian impedance control with damping on
available from http://handbookofrobotics.org/view-chapter/11/videodetails/134

VIDEO 135 Control laws for a single-link arm with an elastic joint
available from http://handbookofrobotics.org/view-chapter/11/videodetails/135

VIDEO 136 Feedforward/feedback law for path tracking with a KUKA KR15/2 robot
available from http://handbookofrobotics.org/view-chapter/11/videodetails/136

VIDEO 770 Trajectory generation and control for a KUKA IR 161/60 robot
available from http://handbookofrobotics.org/view-chapter/11/videodetails/770

VIDEO 777 Input shaping on a lightweight gantry robot
available from http://handbookofrobotics.org/view-chapter/11/videodetails/777

VIDEO 778 Inverse dynamics control for a flexible link
available from http://handbookofrobotics.org/view-chapter/11/videodetails/778

VIDEO 779 Rest-to-rest motion for a flexible link
available from http://handbookofrobotics.org/view-chapter/11/videodetails/779

VIDEO 780 PID response to impulse in presence of link flexibility
available from http://handbookofrobotics.org/view-chapter/11/videodetails/780

VIDEO 781 State feedback response to impulse in presence of link flexibility
available from http://handbookofrobotics.org/view-chapter/11/videodetails/781

References

11.1 L.M. Sweet, M.C. Good: Redefinition of the robot motion control problem, IEEE Control Syst. Mag. **5**(3), 18–24 (1985)

11.2 M.C. Good, L.M. Sweet, K.L. Strobel: Dynamic models for control system design of integrated robot and drive systems, ASME J. Dyn. Syst. Meas. Control **107**, 53–59 (1985)

11.3 E. Rivin: *Mechanical Design of Robots* (McGraw-Hill, New York 1988)

11.4 S. Nicosia, F. Nicolò, D. Lentini: Dynamical control of industrial robots with elastic and dissipative joints, 8th IFAC World Congr., Kyoto (1981) pp. 1933–1939

11.5 P. Tomei: An observer for flexible joint robots, IEEE Trans. Autom. Control **35**(6), 739–743 (1990)

11.6 R. Höpler, M. Thümmel: Symbolic computation of the inverse dynamics of elastic joint robots, IEEE Int. Conf. Robotics Autom. (ICRA), New Orleans (2004) pp. 4314–4319

11.7 M.W. Spong: Modeling and control of elastic joint robots, ASME J. Dyn. Syst. Meas. Control **109**, 310–319 (1987)

11.8 S.H. Murphy, J.T. Wen, G.N. Saridis: Simulation and analysis of flexibly jointed manipulators, 29th IEEE Conf. Decis. Control, Honolulu (1990) pp. 545–550

Avidbots Ex1016
Page 323 of 2154

11.9 R. Marino, S. Nicosia: *On the Feedback Control of Industrial Robots with Elastic Joints: A Singular Perturbation Approach* (Dipartimento di Ingegneria Elettronica, Univ. Rome Tor Vergata, Rome 1984), Rep. R-84.01

11.10 G. Cesareo, F. Nicolò, S. Nicosia: DYMIR: A code for generating dynamic model of robots, IEEE Int. Conf. Robotics Autom. (ICRA), Atlanta (1984) pp. 115–120

11.11 A. De Luca: Feedforward/feedback laws for the control of flexible robots, IEEE Int. Conf. Robotics Autom. (ICRA), San Francisco (2000) pp. 233–240

11.12 G. Buondonno, A. De Luca: A recursive Newton–Euler algorithm for robots with elastic joints and its application to control, IEEE/RSJ Int. Conf. Intell. Robots Syst., Hamburg (2015) pp. 5526–5532

11.13 H.B. Kuntze, A.H.K. Jacubasch: Control algorithms for stiffening an elastic industrial robot, IEEE J. Robotics Autom. **1**(2), 71–78 (1985)

11.14 S.H. Lin, S. Tosunoglu, D. Tesar: Control of a six-degree-of-freedom flexible industrial manipulator, IEEE Control Syst. Mag. **11**(2), 24–30 (1991)

11.15 P. Tomei: A simple PD controller for robots with elastic joints, IEEE Trans. Autom. Control **36**(10), 1208–1213 (1991)

11.16 A. De Luca, B. Siciliano: Regulation of flexible arms under gravity, IEEE Trans. Robotics Autom. **9**(4), 463–467 (1993)

11.17 A. De Luca, B. Siciliano, L. Zollo: PD control with on-line gravity compensation for robots with elastic joints: Theory and experiments, Automatica **41**(10), 1809–1819 (2005)

11.18 C. Ott, A. Albu-Schäffer, A. Kugi, S. Stramigioli, G. Hirzinger: A passivity based Cartesian impedance controller for flexible joint robots – Part I: Torque feedback and gravity compensation, IEEE Int. Conf. Robotics Autom. (ICRA), New Orleans (2004) pp. 2659–2665

11.19 L. Zollo, B. Siciliano, A. De Luca, E. Guglielmelli, P. Dario: Compliance control for an anthropomorphic robot with elastic joints: Theory and experiments, ASME J. Dyn. Syst. Meas. Control **127**(3), 321–328 (2005)

11.20 A. Albu-Schäffer, C. Ott, G. Hirzinger: A passivity based Cartesian impedance controller for flexible joint robots – Part II: Full state feedback, impedance design and experiments, IEEE Int. Conf. Robotics Autom. (ICRA), New Orleans (2004) pp. 2666–2672

11.21 C. Ott, A. Albu-Schäffer, A. Kugi, G. Hirzinger: On the passivity-based impedance control of flexible joint robots, IEEE Trans. Robotics **24**(2), 416–429 (2008)

11.22 A. De Luca, F. Flacco: A PD-type regulator with exact gravity cancellation for robots with flexible joints, IEEE Int. Conf. Robotics Autom. (ICRA), Shanghai (2011) pp. 317–323

11.23 R. Kelly, V. Santibanez: Global regulation of elastic joint robots based on energy shaping, IEEE Trans. Autom. Control **43**(10), 1451–1456 (1998)

11.24 J. Alvarez-Ramirez, I. Cervantes: PID regulation of robot manipulators with elastic joints, Asian J. Control **5**(1), 32–38 (2003)

11.25 A. De Luca, S. Panzieri: Learning gravity compensation in robots: Rigid arms, elastic joints, flexible links, Int. J. Adapt. Control Signal Process. **7**(5), 417–433 (1993)

11.26 A. Albu-Schäffer, G. Hirzinger: A globally stable state feedback controller for flexible joint robots, Adv. Robotics **15**(8), 799–814 (2001)

11.27 L.E. Pfeffer, O. Khatib, J. Hake: Joint torque sensory feedback in the control of a PUMA manipulator, IEEE Trans. Robotics Autom. **5**(4), 418–425 (1989)

11.28 M. Hashimoto, Y. Kiyosawa, R.P. Paul: A torque sensing technique for robots with harmonic drives, IEEE Trans. Robotics Autom. **9**(1), 108–116 (1993)

11.29 T. Lin, A.A. Goldenberg: Robust adaptive control of flexible joint robots with joint torque feedback, IEEE Int. Conf. Robotics Autom. (ICRA), Nagoya (1995) pp. 1229–1234

11.30 M.G. Forrest-Barlach, S.M. Babcock: Inverse dynamics position control of a compliant manipulator, IEEE J. Robotics Autom. **3**(1), 75–83 (1987)

11.31 K.P. Jankowski, H. Van Brussel: An approach to discrete inverse dynamics control of flexible-joint robots, IEEE Trans. Robotics Autom. **8**(5), 651–658 (1992)

11.32 W.M. Grimm: Robustness analysis of nonlinear decoupling for elastic-joint robots, IEEE Trans. Robotics Autom. **6**(3), 373–377 (1990)

11.33 A. De Luca, R. Farina, P. Lucibello: On the control of robots with visco-elastic joints, IEEE Int. Conf. Robotics Autom. (ICRA), Barcelona (2005) pp. 4297–4302

11.34 A. De Luca: Decoupling and feedback linearization of robots with mixed rigid/elastic joints, Int. J. Robust Nonlin. Control **8**(11), 965–977 (1998)

11.35 A. De Luca, B. Siciliano: Inversion-based nonlinear control of robot arms with flexible links, AIAA J. Guid. Control Dyn. **16**(6), 1169–1176 (1993)

11.36 A. De Luca: Dynamic control of robots with joint elasticity, IEEE Int. Conf. Robotics Autom. (ICRA), Philadelphia (1988) pp. 152–158

11.37 S. Nicosia, P. Tomei: On the feedback linearization of robots with elastic joints, 27th IEEE Conf. Decis. Control, Austin (1988) pp. 180–185

11.38 A. De Luca, P. Lucibello: A general algorithm for dynamic feedback linearization of robots with elastic joints, IEEE Int. Conf. Robotics Autom. (ICRA), Leuven (1998) pp. 504–510

11.39 J. Swevers, D. Torfs, M. Adams, J. De Schutter, H. Van Brussel: Comparison of control algorithms for flexible joint robots implemented on a KUKA IR 161/60 industrial robot, 5th Int. Conf. Adv. Robotics, Pisa (1991) pp. 120–125

11.40 K. Khorasani, P.V. Kokotovic: Feedback linearization of a flexible manipulator near its rigid body manifold, Syst. Control Lett. **6**, 187–192 (1985)

11.41 M.W. Spong, K. Khorasani, P.V. Kokotovic: An integral manifold approach to the feedback control of

Avidbots Ex1016
Page 324 of 2154

flexible joint robots, IEEE J. Robotics Autom. **3**(4), 291–300 (1987)
11.42 S. Nicosia, P. Tomei: Design of global tracking controllers for flexible-joint robots, J. Robotic Syst. **10**(6), 835–846 (1993)
11.43 B. Brogliato, R. Ortega, R. Lozano: Global tracking controllers for flexible-joint manipulators: A comparative study, Automatica **31**(7), 941–956 (1995)
11.44 M.W. Spong: Adaptive control of flexible joint manipulators, Syst. Control Lett. **13**(1), 15–21 (1989)
11.45 F. Ghorbel, J.Y. Hung, M.W. Spong: Adaptive control of flexible-joint manipulators, IEEE Control Syst. Mag. **9**(7), 9–13 (1989)
11.46 R. Lozano, B. Brogliato: Adaptive control of robot manipulators with flexible joints, IEEE Trans. Autom. Control **37**(2), 174–181 (1992)
11.47 H. Sira-Ramirez, M.W. Spong: Variable structure control of flexible joint manipulators, Int. Robotics Autom. **3**(2), 57–64 (1988)
11.48 A. De Luca, G. Ulivi: Iterative learning control of robots with elastic joints, IEEE Int. Conf. Robotics Autom. (ICRA), Nice (1992) pp. 1920–1926
11.49 O. Dahl: Path constrained motion optimization for rigid and flexible joint robots, IEEE Int. Conf. Robotics Autom. (ICRA), Atlanta (1993) pp. 223–229
11.50 A. De Luca, L. Farina: Dynamic scaling of trajectories for robots with elastic joints, IEEE Int. Conf. Robotics Autom. (ICRA), Washington (2002) pp. 2436–2442
11.51 S. Haddadin, M. Özparpucu, A. Albu-Schäffer: Optimal control for maximizing potential energy in a variable stiffness joint, IEEE 51st Conf. Decis. Control, Maui (2012) pp. 1199–1206
11.52 M. Özparpucu, S. Haddadin: Optimal control of elastic joints with variable damping, 13th European Control Conf., Strasbourg (2014) pp. 2526–2533
11.53 S. Nicosia, P. Tomei, A. Tornambè: A nonlinear observer for elastic robots, IEEE J. Robotics Autom. **4**(1), 45–52 (1988)
11.54 S. Nicosia, P. Tomei: A method for the state estimation of elastic joint robots by global position measurements, Int. J. Adapt. Control Signal Process. **4**(6), 475–486 (1990)
11.55 A. De Luca, D. Schröder, M. Thümmel: An acceleration-based state observer for robot manipulators with elastic joints, IEEE Int. Conf. Robotics Autom. (ICRA), Rome (2007) pp. 3817–3823
11.56 M.W. Spong: On the force control problem for flexible joint manipulators, IEEE Trans. Autom. Control **34**(1), 107–111 (1989)
11.57 J.K. Mills: Stability and control of elastic-joint robotic manipulators during constrained-motion tasks, IEEE Trans. Robotics Autom. **8**(1), 119–126 (1992)
11.58 K.P. Jankowski, H.A. El Maraghy: Dynamic decoupling for hybrid control of rigid-/flexible-joint robots interacting with the environment, IEEE Trans. Robotics Autom. **8**(5), 519–534 (1992)
11.59 T. Lin, A.A. Goldenberg: A unified approach to motion and force control of flexible joint robots, IEEE Int. Conf. Robotics Autom. (ICRA), Minneapolis (1996) pp. 1115–1120
11.60 A. Albu-Schäffer, C. Ott, G. Hirzinger: A unified passivity-based control framework for position, torque and impedance control of flexible joint robots, Int. J. Robotics Res. **26**(1), 23–39 (2007)
11.61 W. Book, O. Maizza-Neto, D.E. Whitney: Feedback control of two beam, two joint systems with distributed flexibility, ASME J. Dyn. Syst. Meas. Control **97**(4), 424–431 (1975)
11.62 W. Book: Characterization of strength and stiffness constraints on manipulator control. In: *Theory and Practice of Robots and Manipulators*, ed. by W. Book (Elsevier, Amsterdam 1977) pp. 37–45
11.63 T.E. Alberts, W. Book, S. Dickerson: Experiments in augmenting active control of a flexible structure with passive damping, AIAA 24th Aerosp. Sci. Meet., Reno (1986)
11.64 T. Bailey, J.E. Hubbard Jr.: Distributed piezoelectric-polymer active vibration control of a cantilever beam, J. Guid. Control Dyn. **8**(5), 605–611 (1985)
11.65 W. Book, V. Sangveraphunsiri, S. Le: The Bracing Strategy for Robot Operation, Jt. IFToMM-CISM Symp. Theory Robots Manip. (RoManSy), Udine (1984)
11.66 W. Book: Analysis of massless elastic chains with servo controlled joints, ASME J. Dyn. Syst. Meas. Control **101**(3), 187–192 (1979)
11.67 E.C. Pestel, F.A. Leckie: *Matrix Methods in Elastomechanics* (McGraw-Hill, New York 1963)
11.68 R. Krauss: Transfer Matrix Modeling, Ph.D. Thesis (School of Mechanical Engineering, Georgia Institute of Technology, Atlanta 2006)
11.69 W.J. Book: Modeling, Design and Control of Flexible Manipulator Arms, Ph.D. Thesis (Department of Mechanical Engineering, Massachusetts Institute of Technology, Cambridge 1974)
11.70 W. Book: Recursive lagrangian dynamics of flexible manipulators, Int. J. Robotics Res. **3**(3), 87–106 (1984)
11.71 W.J. Book, S.H. Lee: Vibration control of a large flexible manipulator by a small robotic arm, Proc. Am. Control Conf., Pittsburgh (1989) pp. 1377–1380
11.72 J. Lew, S.-M. Moon: A simple active damping control for compliant base manipulators, IEEE/ASME Trans. Mechatron. **2**, 707–714 (1995)
11.73 W.J. Book, J.C. Loper: Inverse dynamics for commanding micromanipulator inertial forces to damp macromanipulator vibration, IEEE, Robot Soc. Int. Conf. Intell. Robots Syst., Kyongju (1999)
11.74 I. Sharf: Active damping of a large flexible manipulator with a short-reach robot, Proc. Am. Control Conf., Seattle (1995) pp. 3329–3333
11.75 L. George, W.J. Book: Inertial vibration damping control of a flexible base manipulator, IEEE/ASME Trans. Mechatron. **8**(2), 268–271 (2003)
11.76 J.F. Calvert, D.J. Gimpel: Method and apparatus for control of system output in response to system input, U.S. Patent 280 1351 (1957)
11.77 O.J.M. Smith: *Feedback Control Systems* (McGraw-Hill, New York 1958)

Avidbots Ex1016
Page 325 of 2154

11.78 N. Singer, W.P. Seering: Preshaping command inputs to reduce system vibration, ASME J. Dyn. Syst. Meas. Control **112**(1), 76–82 (1990)

11.79 W. Singhose, W. Seering, N. Singer: Residual vibration reduction using vector diagrams to generate shaped inputs, J. Mech. Des. **2**, 654–659 (1994)

11.80 D.P. Magee, W.J. Book: The Application of Input Shaping to a System with Varying Parameters, Proc. 1992 Japan-USA Symp. Flex. Autom., San Francisco (1992) pp. 519–526

11.81 D.P. Magee, W.J. Book: Optimal arbitrary time-delay (OAT) filter and method to minimize unwanted system dynamics, U.S. Patent 607 8844 (2000)

11.82 S. Rhim, W.J. Book: Noise effect on time-domain adaptive command shaping methods for flexible manipulator control, IEEE Trans. Control Syst. Technol. **9**(1), 84–92 (2001)

11.83 D.-S. Kwon, W.J. Book: A time-domain inverse dynamic tracking control of a single-link flexible manipulator, J. Dyn. Syst. Meas. Control **116**, 193–200 (1994)

11.84 D.S. Kwon: An Inverse Dynamic Tracking Control for a Bracing Flexible Manipulator, Ph.D. Thesis (School of Mechanical Engineering, Georgia Institute of Technology, Atlanta 1991)

11.85 A. De Luca, G. Di Giovanni: Rest-to-rest motion of a one-link flexible arm, Proc. IEEE/ASME Int. Conf. Adv. Intell. Mechatron., Como (2001) pp. 923–928

11.86 A. De Luca, G. Di Giovanni: Rest-to-rest motion of a two-link robot with a flexible forearm, Proc. IEEE/ASME Int. Conf. Adv. Intell. Mechatron., Como (2001) pp. 929–935

11.87 A. De Luca, V. Caiano, D. Del Vescovo: Experiments on rest-to-rest motion of a flexible arm. In: *Experimental Robotics VIII*, Springer Tracts in Advanced Robotics, Vol. 5, ed. by B. Siciliano, P. Dario (Springer, Berlin, Heidelberg 2003) pp. 338–349

11.88 R.M. Murray, M. Rathinam, W. Sluis: Differential flatness of mechanical control systems: A catalog of prototype systems, Proc. ASME Int. Mech. Engr. Congr., San Francisco (1995)

11.89 M. Fliess, P. Martin, P. Rouchon: Flatness and defect of nonlinear systems: Introductory theory and examples, Int. J. Control **61**(6), 1327–1361 (1995)

11.90 R.H. Cannon, E. Schmitz: Initial experiments on the end-point control of a flexible one-link robot, Int. J. Robotics Res. **3**(3), 62–75 (1984)

11.91 A. Truckenbrot: Modeling and control of flexible manipulator structures, Proc. 4th CISM-IFToMM Symp. Theory Robots Manip. (RoManSy), Zaborow (1981) pp. 90–101

11.92 G.G. Hastings, W.J. Book: Reconstruction and robust reduced-order observation of flexible variables, ASME Winter Ann. Meet., Anaheim (1986)

11.93 B.S. Yuan, J.D. Huggins, W.J. Book: Small motion experiments with a large flexible arm with strain feedback, Proceedings of the 1989 Am. Control Conf., Pittsburgh (1989) pp. 2091–2095

11.94 D. Wang, M. Vidyasagar: Passive control of a stiff flexible link, Int. J. Robotics Res. **11**, 572–578 (1992)

11.95 K. Obergfell, W.J. Book: Control of flexible manipulators using vision and modal feedback, Proc. Int. Conf. Risk Assess. Manag. (ICRAM), Istanbul (1995)

11.96 B. Post: Robust State Estimation for the Control of Flexible Robotic Manipulators, Ph.D. Thesis (School of Mechanical Engineering, Georgia Institute of Technology, Atlanta 2013)

11.97 B. Walcott, S. Zak: State observation of nonlinear uncertain dynamical system, IEEE Trans. Autom. Control **32**, 166–170 (1987)

11.98 B. Walcott, S. Zak: Observation of dynamical systems in the presence of bounded nonlinearities/uncertainties, Proc. IEEE Conf. Dec. Control **25**, 961–966 (1986)

11.99 N.G. Chalhoub, G.A. Kfoury: Development of a robust nonlinear observer for a single-link, flexible manipulator, Nonlin. Dyn. **39**(3), 217–233 (2005)

11.100 W.J. Book: Controlled motion in an elastic world, ASME J. Dyn. Syst. Meas. Control **2B**, 252–261 (1993)

11.101 C. Canudas-de-Wit, B. Siciliano, G. Bastin (Eds.): *Theory of Robot Control* (Springer, Berlin, Heidelberg 1996)

11.102 G. Hirzinger, N. Sporer, J. Butterfass, M. Grebenstein: Torque-controlled lightweight arms and articulated hands: Do we reach technological limits now?, Int. J. Robotics Res. **23**(4/5), 331–340 (2004)

11.103 Camotion Inc.: http://www.camotion.com (Camotion Inc., Atlanta 2007)

11.104 B. Rooks: High speed delivery and low cost from new ABB packaging robot, Ind. Robotics Int. J. **26**(4), 267–275 (1999)

11.105 T. Trautt, E. Bayo: Inverse dynamics of non-minimum phase systems with non-zero initial conditions, Dyn. Control **7**(1), 49–71 (1997)

11.106 B. Siciliano, W. Book: A singular perturbation approach to control of lightweight flexible manipulators, Int. J. Robotics Res. **7**(4), 79–90 (1988)

11.107 F. Ghorbel, M.W. Spong: Singular perturbation model of robots with elastic joints and elastic links constrained by a rigid environment, J. Intell. Robotics Syst. **22**(2), 143–152 (1998)

11.108 J. Guldner, J. Shi, V. Utkin: *Sliding Mode Control in Electromechanical Systems* (Taylor, London 1999)

11.109 J.-H. Ryu, D.-S. Kwon, B. Hannaford: Control of a flexible manipulator with noncollocated feedback: Time-domain passivity approach, IEEE Trans. Robotics **20**(4), 776–780 (2004)

Avidbots Ex1016
Page 326 of 2154

Multimedia Contents

12. Robotic Systems Architectures and Programming

David Kortenkamp, Reid Simmons, Davide Brugali

Robot software systems tend to be complex. This complexity is due, in large part, to the need to control diverse sensors and actuators in real time, in the face of significant uncertainty and noise. Robot systems must work to achieve tasks while monitoring for, and reacting to, unexpected situations. Doing all this concurrently and asynchronously adds immensely to system complexity.

The use of a well-conceived architecture, together with programming tools that support the architecture, can often help to manage that complexity. Currently, there is no single architecture that is best for all applications – different architectures have different advantages and disadvantages. It is important to understand those strengths and weaknesses when choosing an architectural approach for a given application.

This chapter presents various approaches to architecting robotic systems. It starts by defining terms and setting the context, including a recounting of the historical developments in the area of robot architectures. The chapter then discusses in more depth the major types of architectural components in use today – behavioral control (Chap. 13), executives, and task planners (Chap. 14) – along with commonly used techniques for interconnecting connecting those components. Throughout, emphasis will be placed on programming tools and environments that support these architectures. A case study is then presented, followed by a brief discussion of further reading.

12.1 **Overview** 283
12.1.1 Special Needs of Robot Architectures 284
12.1.2 Modularity and Hierarchy 284
12.1.3 Software Development Tools 285
12.2 **History** 285
12.2.1 Subsumption 285
12.2.2 Layered Robot Control Architectures 286
12.3 **Architectural Components** 289
12.3.1 Connecting Components 289
12.3.2 Behavioral Control 291
12.3.3 Executive 292
12.3.4 Planning 294
12.4 **Case Study – GRACE** 296
12.5 **The Art of Robot Architectures** 298
12.6 **Implementing Robotic Systems Architectures** 299
12.6.1 Agile Robot Development Network (aRDnet) 300
12.6.2 Yet Another Robot Platform (YARP) 300
12.6.3 Open Robot Control Software (OROCOS) 300
12.6.4 Smartsoft 300
12.6.5 Robot Operating System (ROS) 301
12.6.6 GenoM/BIP 301
12.6.7 Best Practice in Robotics (BRICS) 301
12.7 **Conclusions and Further Reading** 302
Video-References 302
References 302

12.1 Overview

The term *robot architecture* is often used to refer to two related, but distinct, concepts. Architectural *structure* refers to how a system is divided into subsystems and how those subsystems interact. The structure of

a robot system is often represented informally using traditional *boxes and arrows* diagrams or more formally using techniques such as unified modeling language (UML) [12.1]. In contrast, architectural *style* refers to the computational concepts that underlie a given system. For instance, one robot system might use a publish–subscribe message passing style of communication, while another may use a more synchronous client–server approach.

All robotic systems use some architectural structure and style. However, in many existing robot systems it is difficult to pin down precisely the architecture being used. In fact, a single robot system will often use several styles together. In part, this is because the system implementation may not have clean subsystem boundaries, blurring the architectural structure. Similarly, the architecture and the domain-specific implementation are often intimately tied together, blurring the architectural style(s).

This is unfortunate, as a well-conceived, clean architecture can have significant advantages in the specification, execution, and validation of robot systems. In general, robot architectures facilitate development by providing beneficial constraints on the design and implementation of robotic systems, without being overly restrictive. For instance, separating behaviors into modular units helps to increase understandability and reusability, and can facilitate unit testing and validation.

12.1.1 Special Needs of Robot Architectures

In some sense, one may consider robot architectures as software engineering. However, robot architectures are distinguished from other software architectures because of the special needs of robot systems. The most important of these, from the architectural perspective, are that robot systems need to interact asynchronously, in real time, with an uncertain, often dynamic, environment. In addition, many robot systems need to respond at varying temporal scopes – from millisecond feedback control to minutes, or hours, for complex tasks.

To handle these requirements, many robot architectures include capabilities for acting in real time, controlling actuators and sensors, supporting concurrency, detecting and reacting to exceptional situations, dealing with uncertainty, and integrating high-level (symbolic) planning with low-level (numerical) control.

While the same capability can often be implemented using different architectural styles, there may be advantages of using one particular style over another. As an example, consider how a robot system's style of communications can impact on its reliability. Many robot systems are designed as asynchronous processes that communicate using message passing. One popular communication style is *client–server*, in which a message request from the client is paired with a response from the server. An alternate communication paradigm is *publish–subscribe*, in which messages are broadcast asynchronously and all modules that have previously indicated an interest in such messages receive a copy. With client–server-style message passing, modules typically send a request and then block, waiting for the response. If the response never comes (e.g., the server module crashes) then deadlock can occur. Even if the module does not block, the control flow still typically expects a response, which may lead to unexpected results if the response never arrives or if a response to some other request happens to arrive first. In contrast, systems that use publish–subscribe tend to be more reliable: because messages are assumed to arrive asynchronously, the control flow typically does not assume any particular order in which messages are processed, and so missing or out-of-order messages tend to have less impact.

12.1.2 Modularity and Hierarchy

One common feature of robot architectures is modular decomposition of systems into simpler, largely independent pieces. As mentioned above, robot systems are often designed as communicating processes, where the communications interface is typically small and relatively low bandwidth. This design enables the processes/modules to handle interactions with the environment asynchronously, while minimizing interactions with one another. Typically, this decreases overall system complexity and increases overall reliability.

Often, system decomposition is hierarchical – modular components are themselves built on top of other modular components. Architectures that explicitly support this type of layered decomposition reduce system complexity through abstraction. However, while hierarchical decomposition of robotic systems is generally regarded as a *desirable quality*, debate continues over the dimensions along which to decompose. Some architectures [12.2] decompose along a temporal dimension – each layer in the hierarchy operates at a characteristic frequency an order of magnitude slower than the layer below. In other architectures [12.3–6], the hierarchy is based on task abstraction – tasks at one layer are achieved by invoking a set of tasks at lower levels. In some situations, decomposition based on spatial abstraction may be more useful, such as when dealing with both local and global navigation [12.7]. The main point is that different applications need to decompose

problems in different ways, and architectural styles can often be found to accommodate those different needs.

12.1.3 Software Development Tools

While significant benefit accrues from *designing* systems using well-defined architectural styles, many architectural styles also have associated software tools that facilitate adhering to that style during *implementation*. These tools can take the form of libraries of functions calls, specialized programming languages, or graphical editors. The tools make the constraints of the architectural style explicit, while hiding the complexity of the underlying concepts.

For instance, inter-process communication libraries, such as common object request broker architecture (CORBA) [12.8] and inter-process communication (IPC) package [12.9], make it easy to implement message passing styles, such as client–server and publish–subscribe, respectively. Languages, such as Subsumption [12.10] and Skills [12.11] facilitate the development of data-driven, real-time behaviors, while languages such as the execution support language (ESL) [12.12] and the the plan execution interchange language (PLEXIL) [12.13] provide support for reliably achieving higher-level tasks. Graphical editors, such as found in ControlShell [12.14], Labview [12.15] and open robot controller computer aided design (ORCCAD) [12.6], provide constrained ways to assemble systems, and then automatically generate code that adheres to the architectural style.

In each case, the tools facilitate the development of software in the given style and, more importantly, make it impossible (or, at least, very difficult) to violate the constraints of that architectural style. The result is that systems implemented using such tools are typically easier to implement, understand, debug, verify, and maintain. They also tend to be more reliable, since the tools provide well-engineered capabilities for commonly needed control constructs, such as message passing, interfacing with actuators and sensors, and handling concurrent tasks.

12.2 History

Robot architectures and programming began in the late 1960s with the Shakey robot at Stanford University [12.16] (Fig. 12.1). Shakey had a camera, a range finder, and bump sensors, and was connected to DEC PDP-10 and PDP-15 computers via radio and video links. Shakey's architecture was decomposed into three functional elements: sensing, planning, and executing [12.17]. The sensing system translated the camera image into an internal world model. The planner took the internal world model and a goal and generated a plan (i. e., a series of actions) that would achieve the goal. The executor took the plan and sent the actions to the robot. This approach has been called the *sense–plan–act* (SPA) paradigm (Fig. 12.2). Its main architectural features are that sensing flowed into a world model, which was then used by the planner, and that plan was executed without directly using the sensors that created the model. For many years, robotic control architectures and programming focused almost exclusively on the SPA paradigm.

12.2.1 Subsumption

In the early 1980s, it became apparent that the SPA paradigm had problems. First, planning in any real-world domain took a long time, and the robot would be blocked, waiting for planning to complete. Second, and more importantly, execution of a plan without involving sensing was dangerous in a dynamic world. Several new robot control architecture paradigms be-



Fig. 12.1 Shakey (after [12.18])

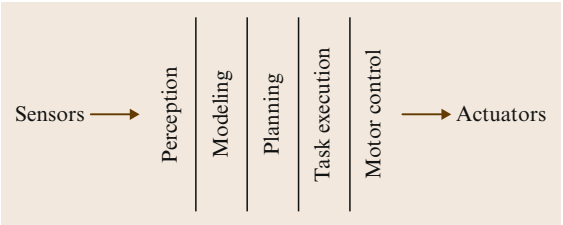


Fig. 12.2 The sense–plan–act (SPA) paradigm (after [12.3], with permission)

Avidbots Ex1016
Page 329 of 2154

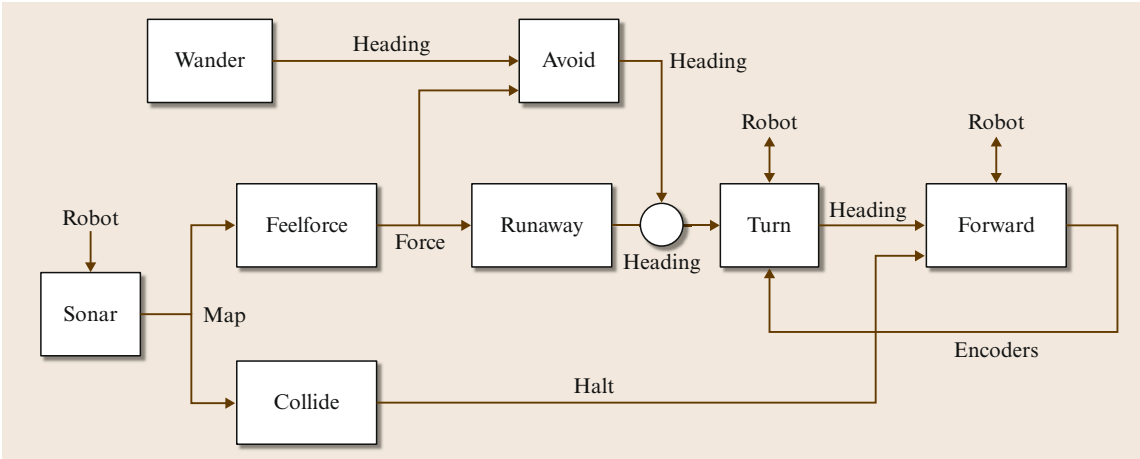


Fig. 12.3 Example of the Subsumption architecture (after [12.3])

gan to emerge, including reactive planning, in which plans were generated quickly and relied more directly on sensed information instead of internal models [12.4, 19]. The most influential work, however, was the subsumption architecture of *Brooks* [12.3]. A subsumption architecture is built from layers of interacting finite-state machines – each connecting sensors to actuators directly (Fig. 12.3). These finite-state machines were called *behaviors* (leading some to call Subsumption *behavior-based* or *behavioral* robotics [12.20]; see also Chap. 38). Since multiple behaviors could be active at any one time, Subsumption had an *arbitration* mechanism that enabled higher-level behaviors to override signals from lower-level behaviors. For example, the robot might have a behavior that simply drives the robot in random directions. This behavior is always active and the robot is always driving somewhere. A second, higher-level behavior could take sensor input, detect obstacles, and steer the robot away from them. It is also always active. In an environment with no obstacles, the higher-level behavior never generates a signal. However, if it detects an obstacle it overrides the lower-level behavior and steers the robot away. As soon as the obstacle is gone (and the higher-level behavior stops sending signals), the lower-level behavior gets control again. Multiple, interacting layers of behaviors could be built to produce more and more complex robots.

Many robots were built using the subsumption approach – most at MIT [12.21–23]. They were quite successful. Whereas SPA robots were slow and ponderous, Subsumption robots were fast and reactive. A dynamic world did not bother them because they constantly sensed the world and reacted to it. These robots scampered around like insects or small rodents. Several behavioral architectures arose in addition to Subsumption, often with different arbitration schemes for combining the outputs of behaviors [12.24, 25].

A popular example of behavior-based architectures is *Arkin*'s motor-control schemas [12.26]. In this biologically inspired approach, motor and perceptual schemas [12.27] are dynamically connected to one another. The motor schemas generate response vectors based on the outputs of the perceptual schemas, which are then combined in a manner similar to potential fields [12.28]. To achieve more complex tasks, the autonomous robot architecture (AuRA) architecture [12.29, 30] added a navigation planner and a plan sequencer, based on finite-state acceptors (FSAs), to the reactive schemas.

However, behavior-based robots soon reached limits in their capabilities. It proved very difficult to compose behaviors to achieve long-range goals and it proved almost impossible to optimize robot behavior. For example, a behavior-based robot that delivered mail in an office building could easily be built by simply wandering around the office building and having behaviors looking for rooms and then overriding the wandering and entering the office. It was much more difficult to use the behavioral style of architecture to design a system that reasoned about the day's mail to visit the offices in an optimal order to minimize delivery time. In essence, robots needed the planning capabilities of the early architectures wedded to the reactivity of the behavior-based architectures. This realization led to the development of layered, or tiered, robot control architectures.

12.2.2 Layered Robot Control Architectures

One of the first steps towards the integration of reactivity and deliberation was the reactive action packages

(RAPs) system created by *Firby*. In his thesis [12.31], we see the first outline of an integrated, three-layer architecture. The middle layer of that architecture, and the subject of the thesis, was the RAPs system. Firby also speculated on the form and function of the other two tiers, specifically with the idea of integrating classic deliberative methods with the ideas of the emerging situated reasoning community, but those layers were never implemented. Later, *Firby* would integrate RAPs with a continuous low-level control layer [12.32].

Independently and concurrently, *Bonasso* at MITRE [12.33] devised an architecture that began at the bottom layer with robot behaviors programmed in the Rex language as synchronous circuits [12.34]. These Rex machines guaranteed consistent semantics between the agent's internal states and those of the world. The middle layer was a conditional sequencer implemented in the goal as parallel programs (GAPPs) language [12.35], which would continuously activate and deactivate the Rex skills until the robot's task was complete. This sequencer based on GAPPs was appealing because it could be synthesized through more traditional planning techniques [12.36]. This work culminated in the three tiers (3T) architecture (named after its three tiers of interacting control processes – planning, sequencing, and real-time control), which has been used on many generations of robots [12.37].

Architectures similar to 3T (Fig. 12.4) have been developed subsequently. One example is a three layer architecture for navigating through intricate situations (ATLANTIS) [12.38], which leaves much more control at the sequencing tier. In ATLANTIS, the deliberative tier must be specifically called by the sequencing tier. A third example is *Saridis*' intelligent control architecture [12.39]. The architecture begins with the servo systems available on a given robot and augments them to integrate the execution algorithms of the next level, using VxWorks and the VME (Versa Module Europa) bus. The next level consists of a set of coordinating routines for each lower subsystem, e.g., vision, arm motion, and navigation. These are implemented in Petri net transducers (PNTs), a type of scheduling mechanism, and activated by a dispatcher connected to the organizational level. The organizational level is a planner implemented as a Boltzmann neural network. Essentially the neural network finds a sequence of actions that will match the required command received as text input, and then the dispatcher executes each of these steps via the network of PNT coordinators.

The LAAS architecture for autonomous systems (LAAS) is a three-layered architecture that includes software tools to support development/programming at each layer [12.40]. The lowest layer (functional) consists of a network of *modules*, which are dynamically parameterized control and perceptual algorithms. Modules are written in the generator of modules (GenoM) language, which produces standardized templates that facilitate the integration of modules with one another. Unlike most other three-layered architectures, the executive layer is fairly simple – it is purely reactive and does no task decomposition. It serves mainly as a bridge – receiving task sequences from the highest layer and selecting and parameterizing tasks to send to the functional layer. The executive is written in the Kheops language, which automatically generates decision networks that can be formally verified. At the top, the decision layer consists of a planner, implemented using the indexed time table (IxTeT) temporal planner [12.41, 42], and a supervisor, implemented using procedural reasoning system (PRS) [12.43, 44]. The supervisor is similar to the executive layer of other three-layered architectures – it decomposes tasks, chooses alternative methods for achieving tasks, and monitors execution. By combining the planner and supervisor in one layer, LAAS achieves a tighter connection between the two, enabling more flexibility in when, and how, replanning occurs. The LAAS architecture actually allows for multiple decisional layers at increasingly higher levels of abstraction, such as a high-level *mission* layer and a lower-level *task* layer.

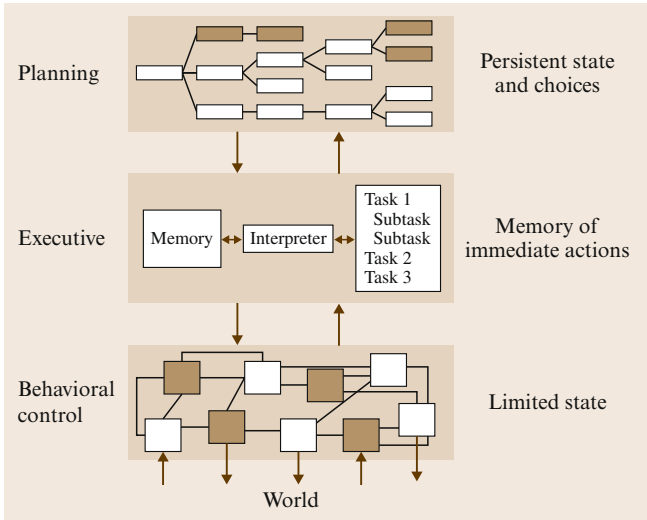


Fig. 12.4 Prototype three-tiered architecture

Remote agent is an architecture for the autonomous control of spacecraft [12.45]. It actually consists of four layers – a control (behavioral) layer, an executive, a planner/scheduler, and mode identification and recovery (MIR) that combines fault detection and recovery. The control layer is the traditional spacecraft real-time control system. The executive is the core of

Avidbots Ex1016
Page 331 of 2154

the architecture – it decomposes, selects, and monitors task execution, performs fault recovery, and does resource management, turning devices on and off at appropriate times to conserve limited spacecraft power. The planner/scheduler is a batch process that takes goals, an initial (projected) state, and currently scheduled activities, and produces plans that include flexible ranges on start and end times of tasks. The plan also includes a task to reinvoke the planner to produce the next plan segment. An important part of the remote agent is configuration management – configuring hardware to support tasks and monitoring that the hardware remains in known, stable states. The role of configuration management is split between the executive, which uses reactive procedures, and MIR, which uses declarative models of the spacecraft and deliberative algorithms to determine how to reconfigure the hardware in response to detected faults [12.46].

The Syndicate architecture [12.47] extends the 3T model to multirobot coordination (Chap. 51). In this architecture, each layer interfaces not only with the layers above and below, as usual, but also with the layers of the other robots at the same level (Fig. 12.5). In this way, distributed control loops can be designed at multiple levels of abstraction. The version of Syndicate in [12.48] used a distributed market-based approach for task allocation at the planning layer.

Other noteworthy multitiered architectures have appeared in the literature. The National Bureau of Standards (NBS) developed for the Aeronautics and Space Agency (NASA) the NASA/NBS standard reference model (NASREM) [12.2, 49], later named real-time control system (RCS), was an early reference model for telerobotic control (Fig. 12.6). It is a many-tiered model in which each layer has the same general structure, but operates at increasingly lower frequency as it moves from the servo level to the reasoning levels. With the exception of maintaining a global world model, NASREM, in its original inception, provided for all the data and control paths that are present in architectures such as 3T, but NASREM was a reference model, not an implementation. The subsequent implementations of NASREM followed primarily the traditional sense–plan–act approach and were mainly applied to telerobotic applications, as opposed to autonomous robots. A notable exception was the early work of *Blidberg/Chappell* [12.50].

While three-layered robot architectures are very popular, various two-layered architectures have been investigated by researchers. The coupled layered architecture for robot autonomy (CLARAty) was designed to provide reusable software for NASA's space robots, especially planetary rovers [12.52, 53]. CLARAty consists of a functional and a decision layer. The functional layer is a hierarchy of object-oriented algorithms that provide more and more abstract interfaces to the robot, such as motor control, vehicle control, sensor-based navigation, and mobile manipulation. Each object provides a generic interface that is hardware independent, so that the same algorithms can run on different hardware. The decision layer combines planning and executive capabilities. Similar to the LAAS architecture,

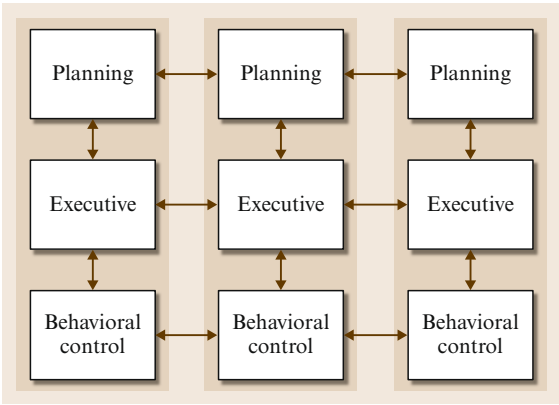


Fig. 12.5 The Syndicate multirobot architecture

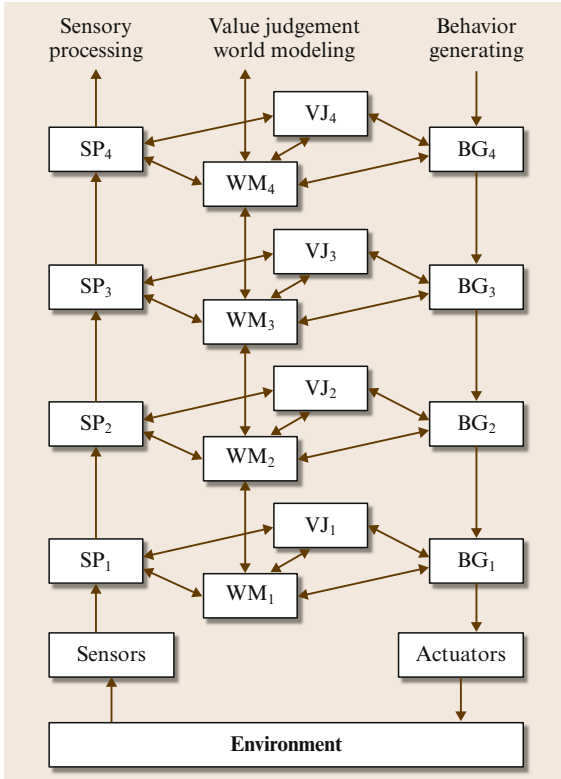


Fig. 12.6 The real-time control system (RCS) reference architecture (after [12.51])

this is done to provide for tighter coordination between planning and execution, enabling continual replanning in response to dynamic contingencies.

Closed-loop execution and recovery (CLEaR) [12.54] is one instantiation of the CLARAty decision layer. CLEaR combines the continuous activity scheduling, planning, execution and replanning (CASPER) repair-based planner [12.55] and the task description language (TDL) executive language [12.56]. CLEaR provides a tightly coupled approach to goal- and event-driven behavior. At its heart is the capability to do fast, continuous replanning, based on frequent state and resource updates from execution monitoring. This enables the planner to react to many exceptional situations, which can be important in cases where there are many tasks, few resources, and significant uncertainty. In CLEaR, both the planning and executive components are able to handle resource conflicts and exceptional situations – heuristics are used to decide which component should be involved in a given situation. The onboard autonomous science investigation system (OASIS) system [12.57] extends CLEaR to include science data analysis so that the architecture can be driven by opportunistic science-related goals (such as finding unusual rocks or formations). OASIS is planner-centric, releasing tasks to the executive component just a few seconds before their scheduled start times.

The cooperative intelligent real-time control architecture (CIRCA) is a two-layered architecture concerned with guaranteeing reliable behavior [12.58, 59]. It embodies the notion of *bounded reactivity* – an acknowledgement that the resources of the robot are not always sufficient to guarantee that all tasks can be achieved. CIRCA consists of a real-time system (RTS) and an artificial intelligence (AI) system (AIS) that are largely independent. The RTS executes a cyclic schedule of test action pairs (TAPs) that have guaranteed worst-case behavior in terms of sensing the environment and conditionally acting in response. It is the responsibility of the AIS to create a schedule that is guaranteed to prevent catastrophic failures from occurring, while running in hard real time. The AIS does this by planning over a state-transition graph that includes transitions for actions, exogenous events, and the passage of time (e.g., if the robot waits too long, bad things can happen). The AIS tests each plan (set of TAPs) to see if it can actually be scheduled. If not, it alters the planning model, either by eliminating tasks (based on goal prioritization) or by changing parameters of behaviors (e.g., reducing the robot's velocity). The AIS continues this until it finds a plan that can be successfully scheduled, in which case it downloads the new plan to the RTS in an atomic operation.

Like CIRCA, ORCCAD is a two-layered architecture that is concerned with guaranteed reliability [12.6, 60]. In the case of ORCCAD, this guarantee is achieved through formal verification techniques. Robot tasks (lower-level behaviors) and robot procedures (higher-level actions) are defined in higher-level languages that are then translated into the Esterel programming language [12.61], for logical verification, or the Timed-Argus language [12.62], for temporal verification. The verification is geared toward liveness and safety properties, as well as verifying lack of contention for resources.

12.3 Architectural Components

We will take the three-tiered architecture as the prototype for the components discussed in this chapter. Figure 12.4 shows a typical three-tiered architecture. The lowest tier (or layer) is behavioral control and is the layer tied most closely to sensors and actuators. The second tier is the executive layer and is responsible for choosing the current behaviors of the robot to achieve a task. The highest tier is the task-planning layer and it is responsible for achieving long-term goals of the robot within resource constraints. Using the example of an office delivery robot, the behavioral layer is responsible for moving the robot around rooms and hallways, for avoiding obstacles, for opening doors, etc. The executive layer coordinates the behavioral layer to achieve tasks such as leaving a room, going to an office, etc. The task-planning layer is responsible for deciding the order of deliveries to minimize time, taking into account delivery priorities, scheduling, recharging, etc. The task-planning layer sends tasks (e.g., exit the room, go to office 110) to the executive. All these tiers need to work together and exchange information. The next section deals with the problem of connecting components to each other. We then discuss each component of the three-tiered prototype architecture in detail.

12.3.1 Connecting Components

All of the architecture components that have been discussed in this chapter need to communicate with each other. They need to both exchange data and send commands. The choice of how components communicate (often called the *middleware*) is one of the most im-

portant and most constraining of the many decisions a robot architecture designer will make. From previous experience, a great deal of the problems and a majority of the debugging time in developing robot architectures have to do with communication between components. In addition, once a communication mechanism is chosen it becomes extremely difficult to change, so early decisions persist for many years. Many developers *roll their own* communication protocols, usually built on top of Unix sockets. While this allows for customization of messages, it fails to take advantage of the reliability, efficiency, and ease of use that externally available communication packages provide. There are two basic approaches to communication: client–server and publish–subscribe.

Client–Server

In a client–server (also called a point-to-point) communication protocol, components talk directly with other components. A good example of this is remote procedure call (RPC) protocols in which one component (the client) can call functions and procedures of another component (the server). A modern, and popular, variation on this is the common object request broker architecture (CORBA). CORBA allows for one component to call object methods that are implemented by another component. All method calls are defined in an interface definition language (IDL) file that is language independent. Every component uses the same IDL to generate code that compiles with component to handle communication. The advantage of this is that, when an IDL file is changed, all components that use that IDL can be recompiled automatically (by using *make* or similar code configuration tools). CORBA object request brokers (ORBs) are available for most major object-oriented languages. Although free ORBs are available, many commercial ORBs offer additional features and support. One disadvantage of CORBA is that it introduces quite a bit of additional code into applications. Some competitors have tried to address this issue, such as the internet communications engine (ICE), which has its own version of an IDL file called the specification language for ICE (SLICE). The biggest advantage of a client–server protocol is that the interfaces are very clearly defined in advance and everyone knows when the interface has changed. Another advantage is that it allows for a distributed approach to communication with no central module that must distribute data. A disadvantage of client–server protocols is that they introduce significant overhead, especially if many components need the same information. It should be noted that CORBA and ICE also have a broadcast mechanism (called an event channel, or the notification service, in CORBA).

Publish–Subscribe

In a publish–subscribe (also called a broadcast) protocol, a component publishes data and any other component can subscribe to that data. Typically, a centralized process routes data between publishers and subscribers. In a typical architecture, most components both publish information and subscribe to information published by other components. There are several existing publish–subscribe middleware solutions. A popular one for robotics is the real-time innovations' (RTI) data distribution service (DDS), formerly the network data distribution service (NDDS) [12.63]. Another popular publish–subscribe paradigm is IPC developed at Carnegie Mellon University [12.9]. Many publish-subscribe protocols are converging on using extensible markup language (XML) descriptions to define the data being published, with the added convenience of transmitting XML over HTTP, which allows for significant interoperability with Web-based applications. Publish–subscribe protocols have a large advantage in being simple to use and having low overhead. They are especially useful when it is unknown how many different components might need a piece of data (e.g., multiple user interfaces). Also, components do not get bogged down with repeated requests for information from many different sources. Publish–subscribe protocols are often more difficult to debug because the syntax of the message is often hidden in a simple string type. Thus problems are not revealed until runtime when a component tries, and fails, to parse an incoming message. Publish–subscribe protocols are also not as readable when it comes to sending commands from one module to another. Instead of calling an explicit method or function with parameters, a command is issued by publishing a message with the command and parameters in it and then having that message be parsed by a subscriber. Finally, publish–subscribe protocols often use a single central server to dispatch messages to all subscribers, providing a single point of failure and potential bottleneck.

JAUS

Recently, a standard has emerged in the defense robotics community not only for a communication protocol but also for definitions of messages that are sent via that communication protocol. The joint architecture for unmanned systems (JAUS) defines a set of reusable messages and interfaces that can be used to command autonomous systems [12.64–66]. These reusable components reduce the cost of integrating new hardware components into autonomous systems. Reuse also allows for components developed for one autonomous system to be used by another autonomous system. JAUS has two components: a *domain model* and a *reference*

architecture. The domain model is a representation of the unmanned systems' functions and information. It contains a description of the system's functional and informational capabilities. The former includes models of the system's maneuvering, navigational, sensing, payload, and manipulation capabilities. The latter includes models of the system's internal data, such as maps and system status. The reference architecture provides a well-defined set of messages. Messages cause actions to commence, information to be exchanged, and events to occur. Everything that occurs in a JAUS system is precipitated by messages. This strategy makes JAUS a component-based, message-passing architecture.

The JAUS reference architecture defines a system hierarchy, as shown in Fig. 12.7. The topology defines the *system* as the collection of vehicles, operator control units (OCUs), and infrastructure necessary to provide the full robotic capability. Subsystems are individual units (e.g., vehicles or OCUs) in the system. Nodes define a distinct processing capability within the architecture and route JAUS messages to components. Components provide the different execution capabilities and respond directly to command messages. Components might be sensors (e.g., a SICK laser or a vision sensor), actuators (a manipulator or a mobile base) or payloads (weapons or task sensors). The topology (the layout of particular system, subsystems, nodes, and components) is defined by the system implementers based on task requirements.

At the core of JAUS is a set of well-defined messages. JAUS supports the following message types.

- *Command*: Initiate mode changes or actions
- *Query*: Used to solicit information from a component
- *Inform*: Response to a query
- *Event set up*: Passes parameters to set up an event
- *Event notification*: Sent when the event happens

JAUS has about 30 predefined messages that can be used to control robots. There are messages for control of a robotic vehicle. For example, the *global vector driver* message performs closed-loop control of the desired global heading, altitude, and speed of a mobile vehicle. There are also sensor messages such as *global pose sensor*, which distributes the global position and orientation of the vehicle. There are also manipulation messages in JAUS. For example, the *set joint positions* message sets the desired joint position values. The *set tool point* message specifies the coordinates of the end-effector tool point in terms of the coordinate system attached to the end-effector.

JAUS also has user-definable messages. Messages have headers that follow a specific format and include message type, destination address (e.g., system, subsystem, node, and component), priority, etc. While JAUS is primarily point to point, JAUS messages can also be marked as *broadcast* and distributed to all components. JAUS also defines coordinate systems for navigation and manipulation to ensure all components understand any coordinates sent to them.

12.3.2 Behavioral Control

Behavioral control represents the lowest level of control in a robot architecture. It directly connects sensors and actuators. While these are typically hand-crafted functions written in C or C++, there have been specialized languages developed for behavioral control, including ALFA [12.67], Behavioral Language [12.68], and Rex [12.69]. It is at this level that traditional control theory (e.g., proportional–integral–derivative (PID) functions, Kalman filters, etc.) resides. In architectures such as 3T, the behavioral layer functions as a *Brooksian machine* – that is, the layer is composed of a small number of behaviors (also called skills) that perceive the environment and carry out the actions of the robot.

Example

Consider an office delivery robot that operates in a typical office building. The behavioral control layer con-

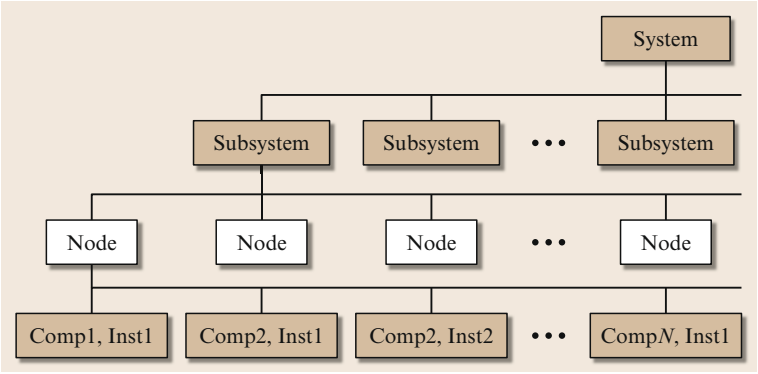


Fig. 12.7 The JAUS reference architecture topology (after JAUS Reference Architecture document [12.64])

Avidbots Ex1016
Page 335 of 2154

tains the control functions necessary to move around in the building and carry out delivery tasks. Assuming the robot has an a priori map of the building, some possible behaviors for this robot include:

1. Move to location while avoiding obstacles
2. Move down hallway while avoiding obstacles
3. Find a door
4. Find a door knob
5. Grasp a door knob
6. Turn a door knob
7. Go through door
8. Determine location
9. Find office number
10. Announce delivery.

Each of these behaviors ties sensors (vision, range sensing, etc.) to actuators (wheel motors, manipulator motors, etc.) in a tight loop. In architectures such as Subsumption, all behaviors are running concurrently with a hierarchical control scheme inhibiting the outputs of certain behaviors. In AuRA [12.30], behaviors are combined using potential functions. Other architectures [12.25, 69] use explicit *arbitration* mechanisms to choose amongst potentially conflicting behaviors.

In architectures such as 3T [12.37], not all of the behaviors are active at the same time. Typically, only a few behaviors that do not conflict would be active at a time (e.g., behaviors 2 and 9 in the example above). The executive layer (Sect. 12.3.3) is responsible for activating and deactivating behaviors to achieve higher-level tasks and to avoid conflicts between two behaviors competing for the same resource (e.g., an actuator).

Situated Behaviors

An important aspect of these behaviors is that they be *situated*. This means that the behavior works only in very specific situations. For example, behavior 2 above moves down a hallway, but this is appropriate only when the robot is situated in a hallway. Similarly, behavior 5, which grasps a door knob, is appropriate only when the robot is within grasping distance of a door knob. The behavior is not responsible for putting the robot in the particular situation. However, it should recognize that the situation is not appropriate and signal as such.

Cognizant Failure

A key requirement for behaviors is that they know when they are not working. This is called *cognizant failure* [12.70]. For example, behavior 5 in our example (grasping the door knob) should not continually grasp at air if it is failing. More succinctly, the behavior should not continue to *bang its head against the wall*. A common problem with early Subsumption robots is that the behaviors did not know they were failing and continued to take actions that were not resulting in progress. It is not the job of the behavioral control layer to decide what to do in a failure situation; it is only necessary to announce that the behavior has failed and halt activity.

Implementation Constraints

The behavioral control layer is designed to bring the speed and reactivity of Subsumption to robot control. For this reason, the behaviors in the behavioral control layer need to follow the philosophies of Subsumption. In particular, the algorithms used for behaviors should be constant in state and time complexity. There should be little or no search at the behavioral control level, and little iteration. Behaviors should simply be transfer functions that take in signals (from sensors or other behaviors) and send out signals (to actuators or other behaviors), and repeat these several times a second. This will allow for reactivity to changing environments. More controversial is how much state should be allowed at the behavioral level. *Brooks* famously said several years ago to *use the world as its own best model* [12.68] – that is, instead of maintaining internal models of the world and querying those models, the robot should instead directly sense the world to get its data. State such as maps, models, etc. belong at the higher levels of the three-tiered prototype architecture, not at the behavioral control layer. Certain exceptions, such as maintaining state for data filtering calculations, could be made on a case-by-case basis. *Gat* [12.71] argues that any state kept at the behavioral layer should be ephemeral and limited.

12.3.3 Executive

The executive layer is the interface between the numerical behavioral control and the symbolic planning layers. The executive is responsible for translating high-level plans into low-level behaviors, invoking behaviors at the appropriate times, monitoring execution, and handling exceptions. Some executives also allocate and monitor resource usage, although that functionality is more commonly performed by the planning layer.

Example

Continuing the example of an office delivery robot, the main high-level task would be to deliver mail to a given office. The executive would decompose this task into a set of subtasks. It may use a geometric path planner to determine the sequence of corridors to move down and intersections at which to turn. If there are doorways along the route, a task would be inserted to open and pass through the door. At the last corridor, the executive would add a concurrent task that looks for the

office number. The final subtasks would be to announce that the person has mail and to concurrently monitor whether the mail has been picked up. If it is not picked up after some period of time, an exception would be triggered that invokes some recovery action (perhaps announcing again, perhaps checking to make sure the robot is at the correct office, perhaps notifying the planning layer to reschedule the delivery for a later time).

Capabilities

The example above illustrates many of the capabilities of the executive layer. First, the executive decomposes high-level tasks (goals) into low-level tasks (behaviors). This is typically done in a *procedural* fashion: the knowledge encoded in the executive describes *how* to achieve tasks, rather than describing *what* needs to be done and having the executive figure out the *how* by itself. Sometimes, though, the executive may also use specialized planning techniques, such as the route planner used in the example above. The decomposition is typically a hierarchical *task tree* (Fig. 12.8), with the leaves of the task tree being invocations and parameterizations of behaviors.

Besides decomposing tasks into subtasks, executives add and maintain temporal constraints between tasks (usually between sibling tasks only, but some executive languages permit temporal constraints between any pair of tasks). The most common constraints are *serial* and *concurrent*, but most executives support more expressive constraint languages, such as having one task begin 10 s after another one starts or having one task end when another ends.

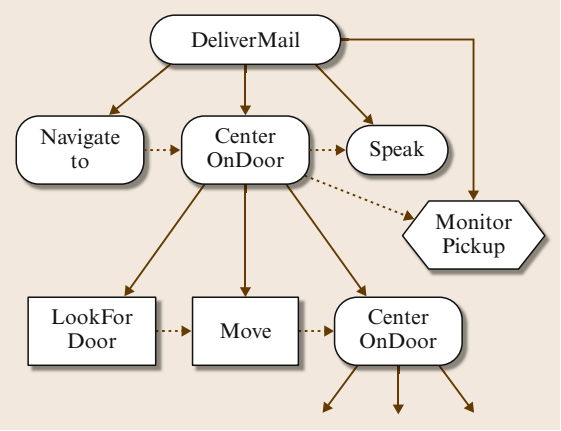


Fig. 12.8 Hierarchical task tree for mail-delivery task (*lozenge nodes* are interior; *rectangular nodes* are leaves; *hexagonal node* is an execution monitor; *solid arrows* are parent–child relationships; *dashed arrows* are sequential constraints)

The executive is responsible for dispatching tasks when their temporal constraints are satisfied. In some executives, tasks may also specify resources (e.g., the robot's motors or camera) that must be available before the task can be dispatched. As with behaviors, arbitrating between conflicting tasks can be a problem. In the case of executives, however, this arbitration is typically either programmed in explicitly (e.g., a rule that says what to do in cases where the robot's attempt to avoid obstacles takes it off the preferred route) or handled using priorities (e.g., recharging is more important than mail delivery).

The final two important executive capabilities are execution monitoring and error recovery. One may wonder why these capabilities are needed if the underlying behaviors are reliable. There are two reasons. First, as described in Sect. 12.3.2, the behaviors are situated, and the situation may change unexpectedly. For instance, a behavior may be implemented assuming that a person is available to pick up the mail, but that may not always be the case. Second, in trying to achieve some goal, the behavior may move the robot into a state that is unexpected by the executive. For instance, people may take advantage of the robot's obstacle avoidance behavior to *herd* it into a closet. While the behavior layer may, in fact, keep the robot safe in such situations, the executive needs to detect the situation in order to get the robot back on track.

Typically, execution monitoring is implemented as a concurrent task that either analyzes sensor data directly or activates a behavior that sends a signal to the executive when the monitored situation arises. These correspond to *polling* and *interrupt-driven* monitors, respectively.

Executives support various responses to monitors being triggered. A monitor may spawn subtasks that handle the situation, it may terminate already spawned subtasks, it may cause the parent task to fail, or it may raise an exception. The latter two responses involve the *error recovery* (also called *exception handling*) capability. Many executives have tasks return status values (success or failure) and allow parent tasks to execute conditionally based on the return values. Other executives use a hierarchical exception mechanism that throws named exceptions to ancestor nodes in the task tree. The closest task that has registered a handler for that exception tries to handle it; if it cannot, it rethrows the exception up the tree. This mechanism, which is inspired by the exception handling mechanisms of C++, Java, and Lisp, is strictly more expressive than the return-value mechanism, but it is also much more difficult to design systems using that approach, due to the nonlocal nature of the control flow.

Avidbots Ex1016
Page 337 of 2154

Implementation Constraints

The underlying formalism for most executives is a hierarchical finite-state controller. Petri nets [12.72] are a popular choice for representing executive functions. In addition, various languages have been developed specifically to assist programmers in implementing executive-level capabilities. We briefly discuss aspects of several of these languages: reactive action packages (RAPs) [12.4, 31], the procedural reasoning system (PRS) [12.43, 44], the execution support language (ESL) [12.12], the task description language (TDL) [12.56], and the plan execution interchange language (PLEXIL) [12.13].

These languages all share features and exhibit differences. One distinction is whether the language is stand-alone (RAPs, PRS, PLEXIL) or an extension of an existing language (ESL is an extension of Common Lisp; TDL is an extension of C++). Stand-alone languages are typically easier to analyze and verify, but extensions are more flexible, especially with respect to integration with legacy software. While stand-alone executive languages all support interfaces to user-defined functions. These interfaces are usually limited in capability (such as what types of data structures can be passed around).

All of these executive languages provide support for hierarchical decomposition of tasks into subtasks. All except PLEXIL allow for recursive invocation of tasks. RAPs, TDL, and PLEXIL have syntax that distinguishes leaf nodes of the task tree/graph from interior nodes.

All these languages provide capabilities for expressing conditionals and iteration, although with RAPs and PLEXIL these are not core-language constructs, but must be expressed as combinations of other constructs. Except for TDL, the languages all provide explicit support for encoding pre- and post-conditions of the tasks and for specifying success criteria. With TDL, these concepts must be programmed in, using more primitive constructs. The stand-alone languages all enable local variables to be defined within a task description, but provide for only limited computation with those variables. Obviously, with extension languages the full capability of the base language is available for defining tasks.

All the languages support the simple serial (sequential) and concurrent (parallel) temporal constraints between tasks, as well as timeouts that can be specified to trigger after waiting a specified amount of time. In addition, TDL directly supports a wide range of temporal constraints – one can specify constraints between the start and end times of tasks (e.g., *task B starts after task A starts* or *task C ends after task D starts*) as well as metric constraints (e.g., *task B starts 10 seconds after task A ends* or *task C starts at 1pm*). ESL and PLEXIL support the signaling of events (e.g., when tasks transition to new states) that can be used to implement similarly expressive types of constraints. In addition, ESL and TDL support task termination based on the occurrence of events (e.g., *task B terminates when task A starts*).

The languages presented differ considerably in how they deal with execution monitoring and exception handling. ESL and TDL both provide explicit execution monitoring constructs and support exceptions that are *thrown* and then *caught* by registered handlers in a hierarchical fashion. This type of exception handling is similar to that used in C++, Java, and Lisp. ESL and TDL also support *clean-up* procedures that can be invoked when tasks are terminated. RAPs and PLEXIL use return values to signal failure, and do not have hierarchical exception handling. PLEXIL, though, does support clean up procedures that are run when tasks fail. PRS has support for execution monitoring, but not exception handling. ESL and PRS support the notion of *resources* that can be shared. Both provide support for automatically preventing contention amongst tasks for the resources. In the other executive languages, this must be implemented separately (although there are plans to extend PLEXIL in this area).

Finally, RAPs, PRS and ESL all include a symbolic database (*world model*) that connects either directly to sensors or to the behavior layer to maintain synchrony with the real world. Queries to the database are used to determine the truth of preconditions, to determine which methods are applicable, etc. PLEXIL has the concept of a *lookup* that performs a similar function, although it is transparent to the task how this is implemented (e.g., by a database lookup or by invoking a behavior-level function). TDL leaves it up to the programmer to specify how the tasks connect to the world.

12.3.4 Planning

The planning component of our prototype layered architecture is responsible for determining the long-range activities of the robot based on high-level goals. Where the behavioral control component is concerned with the here-and-now and the executive is concerned with what has just happened and what should happen next, the planning component looks towards the future. In our running example of an office delivery robot, the planning component would look at the day's deliveries, the resources of the robot, and a map, and determine the optimal delivery routes and schedule, including when the robot should recharge. The planning component is also responsible for replanning when the situation changes.

For example, if an office is locked, the planning component would determine a new delivery schedule that puts that office's delivery later in the day.

Types of Planning

Chapter 9 describes approaches to robot planning in detail. Here, we summarize issues with respect to different types of planners as they relate to layered architectures.

The two most common approaches used are hierarchical task net (HTN) planners and planner/schedulers. HTN planners [12.73, 74] decompose tasks into subtasks, in a manner similar to what many executives do. The main differences are that HTN planners typically operate at higher levels of abstraction, take resource utilization into account, and have methods for dealing with conflicts between tasks (e.g., tasks needing the same resources, or one task negating a precondition needed by another task). The knowledge needed by HTN planners is typically fairly easy to specify, since one indicates directly *how* tasks are to be achieved.

Planner/schedulers [12.75, 76] are useful in domains where time and resources are limited. They create high-level plans that schedule when tasks should occur, but typically leave it to the executive to determine exactly how to achieve the tasks. Planner/schedulers typically work by laying out tasks on time lines, with separate time lines for the various resources that are available on the robot (motors, power, communication, etc.). The knowledge needed by planner/schedulers includes the goals that tasks achieve, the resources they need, their duration, and any constraints between tasks.

Many architectures provide for specialized planning *experts* that are capable of solving particular problems efficiently. In particular, these include motion planners, such as path planners and trajectory planners. Sometimes, the planning layer of the architecture invokes these specialized planners directly; in other architectural styles, the motion planners are part of the lower levels of the architecture (the executive, or even the behavioral layer). Where to put these specialized planners is often a question of style and performance (Sect. 12.5).

Additionally, some architectures provide for multiple planning layers [12.40, 45, 77]. Often, there is a *mission* planning layer at the very top that plans at a very abstract level, over relatively long periods of time. This layer is responsible mainly for selecting which high-level goals are to be achieved over the next period of time (and, in some cases, determining in which order to achieve them) in order to maximize some objective function (e.g., net reward). The lower *task* planning layer is then responsible for determining exactly how and when to achieve each goal. This breakdown is usually done for efficiency reasons, since it is difficult to plan simultaneously at both a detailed level and over a long time horizon.

Integrating Planning and Execution

There are two main approaches to the integration of the planning and execution components in robotic architectures. The first approach is that the planning component is invoked as needed by the executive and returns a plan. The planning component is then dormant until called again. Architectures such as ATLANTIS [12.71] and Remote Agent [12.45] use this approach, which requires that the executive either leave enough time for planning to complete or that it *safes* the system until planning is complete. In the Remote Agent, for instance, a special *planning* task is explicitly scheduled.

The second approach is that the planning component sends high-level tasks down to the executive as required and monitors the progress of those tasks. If tasks fail, replanning is done immediately. In this approach, the planning component is always running and always planning and replanning. Signals must pass in real time between the planner and the executive to keep them synchronized. Architectures such as 3T [12.37] use this second approach. The first approach is useful when the system is relatively static, so that planning can occur infrequently, at relatively predictable times. The second approach is more suited to dynamic environments, where replanning is more frequent and less predictable.

Other decisions that need to be made when integrating planning and execution are when to stop task decomposition, where to monitor plan execution, and how to handle exceptions. By planning all the way down to primitive actions/behaviors, the planner has a very good notion of what will happen during execution, but at a price of much more computation. Also, some task decompositions are easier to describe procedurally (using an executive language) rather than declaratively (using a planning language). Similarly, monitoring at the executive level tends to be more efficient, since the monitoring happens closer to the robot sensors, but the planner may be able to use its more global knowledge to detect exceptions earlier and/or more accurately. With respect to handling exceptions, executives can handle many on their own, at the price of breaking the expectations used by the planner in scheduling tasks. On the other hand, having exceptions handled by the planner typically involves replanning, which can be computationally expensive.

For all these integration issues, however, a middle ground usually exists. For instance, one can choose to decompose some tasks more deeply than others, or han-

Avidbots Ex1016
Page 339 of 2154

dle certain exceptions in the executive and others in the planner. In general, the *right* approach usually involves a compromise and is determined by analyzing the domain in detail (Sect. 12.5).

12.4 Case Study – GRACE

In this section, we present the architecture of a fairly complex autonomous mobile robot. Graduate robot attending conference (GRACE) resulted from the efforts of five research institutions (Carnegie Mellon, Naval Research Laboratory, Northwestern University, Metrica, and Swarthmore College) to tackle the American Association for Artificial Intelligence (AAAI) Robot Challenge. The Challenge was for a robot to attend the AAAI National Conference on Artificial Intelligence as a participant – the robot must find the registration desk (without knowing the layout of the convention center beforehand), register for the conference, and then, after being provided with a map, find its way to a given location in time to give a technical talk about itself.

The architectural design of the robot was particularly important given the complexity of the task and the need to integrate techniques that had been previously developed by the five institutions. These techniques included localization in a dynamic environment, safe navigation in the presence of moving people, path planning, dynamic replanning, visual tracking of people, signs and landmarks, gesture and face recognition, speech recognition and natural language understanding, speech generation, knowledge representation, and social interaction with people.

GRACE is built on top of an real world interface (RWI) B21 base and has an expressive computer-animated face projected on a flat-panel liquid-crystal display (LCD) screen (Fig. 12.9). Sensors that come with the B21 include touch, infrared, and sonar sensors.



Fig. 12.9 The robot GRACE

Near the base is a SICK scanning laser range finder that provides a 180° field of view. In addition, GRACE has several cameras, including a stereo camera head on a pan–tilt unit (PTU) built by Metrica TRACLabs and a single-color camera with pan–tilt–zoom capability, built by Canon. GRACE can speak using a high-quality speech-generation software (Festival), and receive speech responses using a wireless microphone headset (a Shure TC computer wireless transmitter/receiver pair).

The behavioral layer of the architecture consisted of individual processes that controlled particular pieces of hardware. These programs provided abstract interfaces to either control the hardware or return information from sensors. To accommodate the different coding styles of the various groups involved, both synchronous, blocking and asynchronous, nonblocking calls were supported by most of the interfaces (for the nonblocking calls, the interfaces allowed programmers to specify a callback function to be invoked when data was returned). Interfaces at the behavioral level included robot motion and localization (this interface also provided laser information), speech recognition, speech generation, facial animation, color vision, and stereo vision (Fig. 12.10).

The architecture used individual processes for each of the behavioral capabilities, mainly because the underlying code had been developed by different organizations. While having a large number of processes run concurrently is somewhat inefficient, trying to integrate everything into a monolithic process was thought to be too difficult. In addition, the use of separate processes facilitated development and debugging, since one needed to run only those aspects of the system that were being tested.

The executive layer consisted of separate programs for achieving each subtask of the challenge – finding the registration desk, riding the elevator, standing in line, interacting with the person at the desk, navigating to the talk, and giving the talk (Fig. 12.10). As is common in many implemented robotic systems, the GRACE architecture did not have a planning layer – since the high-level plan was fixed and relatively straightforward, it was coded explicitly. Several of the executive-layer programs were written using TDL (Sect. 12.3.3), which facilitated concurrent control and monitoring of the various tasks.

Avidbots Ex1016
Page 340 of 2154

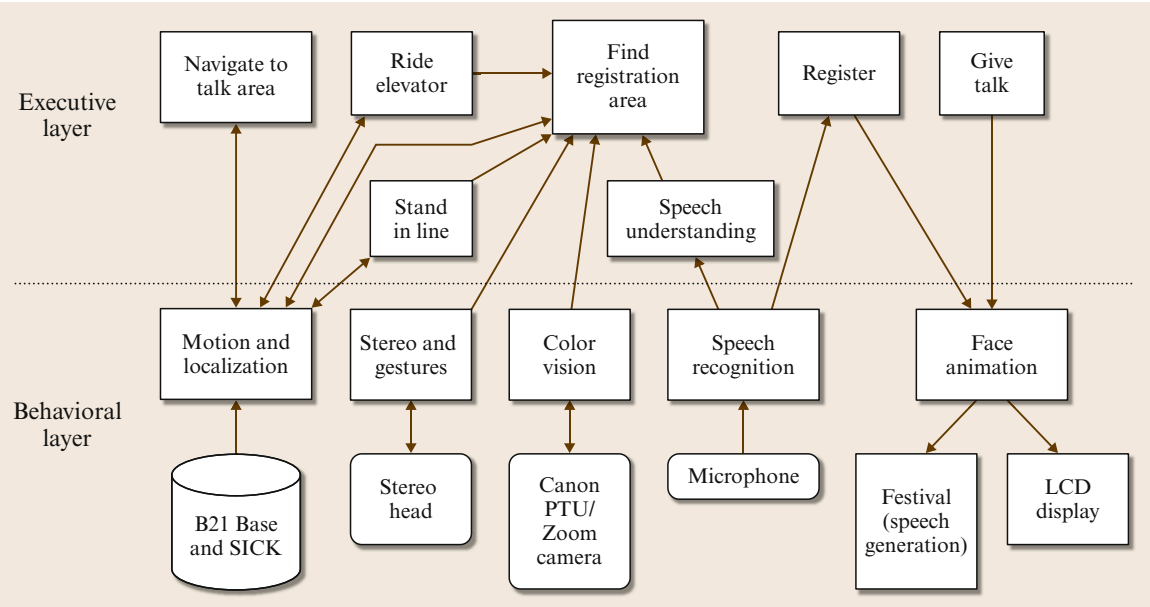


Fig. 12.10 GRACE's architectural structure

One particularly involved task was finding the registration desk (recall that GRACE had no idea where the booth was, or even what the convention center looked like). TDL was used to create a finite-state machine that allowed GRACE to maintain multiple goals, such as using an elevator to get to a particular floor and following directions to find the elevator (Fig. 12.11). The top-level goal was to find the registration desk. Intermediate subgoals were created as GRACE interacted with people to determine the directions to the desk. If there were no directions to follow, GRACE performed a random walk until a person was detected using its laser scanner. GRACE then engaged in conversation with the person to obtain directions. GRACE could handle simple commands, such as *turn left* and *go forward five meters*, as well as higher-level instructions, such as *take the elevator* and *turn left at the next intersection*. In addition, GRACE could ask questions, such as *am I at the registration desk?* and *is this the elevator?* The TDL-based finite-state machine was used to determine which interactions were appropriate at various times and to prevent the robot from getting confused.

Communication between processes used the interprocess communication (IPC) messaging package [12.9, 78]. IPC supports both publish–subscribe and client–server messaging, and enables complex data structures to be passed transparently between processes. One side benefit of using IPC to communicate between processes was the ability to log all message traffic (both message name and data content). This proved invaluable, at times, in determining why the system failed

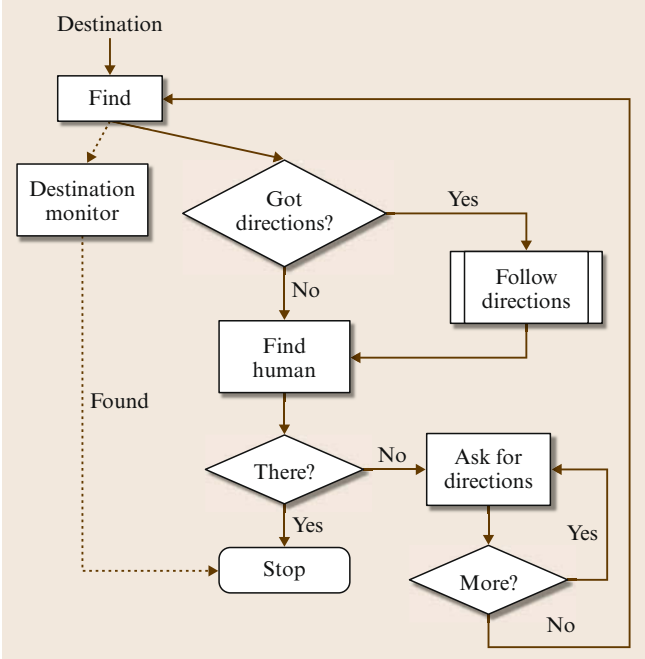


Fig. 12.11 Finite-state machine for GRACE's task for following directions to the registration booth

to act as expected – did a process send out a message with invalid data? Did it fail to send out a message in a timely fashion? Was the receiving process blocked, for some reason? Was there a timing issue? While wading through the message traffic was often tedious, in

Avidbots Ex1016
Page 341 of 2154

some cases it was the only way to catch intermittent bugs.

In July 2002, GRACE successfully completed the challenge at the Shaw Convention Centre in Edmonton, Canada. The processes at the behavioral level generally worked as anticipated – this was largely attributed to the fact that those modules were ported from previously developed (and hence well-tested) systems. While generally functional, the executive-level processes had more problems with off-nominal situations. This is largely attributed to problems in sensor interpretation, as well as mistaken assumptions about what the convention center was going to look like (for instance, it turned out that some of the barriers were made of glass, which is largely invisible to the laser). Overall, however, the architecture itself worked as expected, enabling a large body of complex software to be integrated rather quickly and operate together effectively.

12.5 The Art of Robot Architectures

Designing a robot architecture is much more of an art than a science. The goal of an architecture is to make programming a robot easier, safer, and more flexible. Thus, the decisions made by a developer of a robot architecture are influenced by their own prior experiences (e.g., what programming languages they are familiar with), their robot and its environment, and the tasks that need to be performed. The choice of a robot architecture should not be taken lightly, as it is the authors' experiences that early architectural decisions often persist for years. Changing robot architectures is a difficult proposition and can set back progress while a great deal of code is reimplemented.

The art of designing a robotic architecture starts with a set of questions that the designer needs to ask. These questions include:

- What are the tasks the robot will be performing? Are they long-term tasks? Short-term? User-initiated? Robot-initiated? Are the tasks repetitive or different across time?
- What actions are necessary to perform the tasks? How are those actions represented? How are those actions coordinated? How fast do actions need to be selected/changed? At what speed do each of the actions need to run in order to keep the robot safe?
- What data is necessary to do the tasks? How will the robot obtain that data from the environment or from the users? What sensors will produce the data? What representations will be used for the data? What processes will abstract the sensory data into representations internal to the architecture? How often does the data need to be updated? How often can it be updated?
- What computational capabilities will the robot have? What data will these computational capabilities produce? What data will they consume? How will the computational capabilities of a robot be divided, structured, and interconnected? What is the best decomposition/granularity of computational capabilities? How much does each computational capability have to know about the other capabilities? Are there legacy computational capabilities (from other robots, other robot projects, etc.) that will be used? Where will the different computational capabilities reside (e.g., onboard or offboard)?
- Who are the robot's users? What will they command the robot to do? What information will they want to see from the robot? What understanding do they need of the robot's computational capabilities? How will the user know what the robot is doing? Is the user interaction peer to peer, supervisory, or as a bystander?
- How will the robot be evaluated? What are the success criteria? What are the failure modes? What is the mitigation for those failure modes?
- Will the robot architecture be used for more than one set of tasks? For more than one kind of robot? By more than one team of developers?

Once designers have answers to all (or most) of these questions, they can then begin building some *use cases* for the types of operations they want the robot to perform and how they want users to interact with it. These use cases should specify the outward behavior of the robot with respect to its environment and its users. From the use cases, an initial partitioning of robot functionality can be developed. This partitioning should be accompanied by a *sequence diagram* that shows the transfer of information and control over time amongst the various components of the robot architecture [12.79]. After this, a more formal specification of the interfaces between architectural components can be developed. This may be done using a language such as the interface definition language (IDL) of CORBA or by defining the messages to be distributed in a publish–subscribe protocol. This is an important step, as once implementation begins it is very costly to change interfaces. If an interface does change, all stakeholders need

to be notified and need to agree to the change. The most common integration problems in robot architectures are mismatches between what components expect and what they are receiving in the way of data.

An advantage of tiered architectures with clear interface definitions is that the different layers can be developed in parallel. The behavioral control layer can be implemented and tested on the robot using a human as an executive. The executive can be implemented and tested using state machine *stubs* for the expected behaviors on the robot. The planning layer can be implemented and tested using *stubs* for the tasks in the executive. The *stubs* merely acknowledge that they were called and report back appropriately. Then, the tiers can be integrated to test timing and other runtime issues. This parallel approach speeds up the development of a robot architecture, but is possible only if the roles and interfaces between components are clearly defined and respected. There is still considerable real-time debugging necessary during integration. In our experiences, most of the development time in robot architectures is still spent on the behavioral control layer – that is, sensing and acting are still the hard parts of robot control, as compared to execution and planning. Having a good, robust behavioral control layer goes a long way towards having a competent robot architecture.

12.6 Implementing Robotic Systems Architectures

Robot System Architectures offer a coarse-gain view of the structure of a robot system, which is modelled according to the functional decomposition (or composition) of parts. Unfortunately, there are concerns that cannot be effectively described and modularised using the concept of composition of high cohesion functional modules. Such concerns relate to the software systems as a whole hence crosscutting their modular structure. Non-functional requirements such as real-time performance, fault tolerance, and safety, are typical examples of properties that emerge from multiple parts of a system and cannot be confined into individual modules. In order to meet these requirements, software development efforts are mainly focused on delivering highly efficient implementations of control applications that exploit at best the capabilities of specific hardware platforms to perform a specific set of tasks in specific operational environments.

The race towards performance and the pressure to develop proof-of-concept code to test a new theory [12.80] push robotic engineers to neglect other quality attributes of a software system, such as maintainability, interoperability, and reusability. As a result, a huge corpus of software applications, which implement the entire spectrum of robot functionality, algorithms, and control paradigms, is potentially available in robotic research laboratories as open source libraries. Unfortunately, they are often not reusable even in slightly different application scenarios, because the assumptions and constraints about the computational and robotics hardware, and the software and operational environments are hidden and hard coded in the software implementation.

A large variety of today's robotic systems is designed according to the relatively small number of robot control architecture paradigms discussed in the previous sections. Nevertheless, the software that implements their control applications differs significantly from system to system. These differences relate, for example, to the data structures defined to store relevant information (e.g., the map of the environment, the robot kinematic model), the application programming interfaces (APIs) to drive sensors and actuators [12.81], the information model used to represent key concepts (e.g., geometric relations and coordinate representations). These differences indicate that the most common robot functionalities have been re-implemented from scratch innumerable times.

During the last few years, many ideas from software engineering (such as component-based development and model-driven engineering) have been progressively introduced in the construction of robotic software systems to simplify their development and improve their quality (see [12.82] for a survey).

Modern robot control systems are typically designed as (logically) distributed component-based systems [12.83, 84]. Here, components are units of implementation and represent a code-based way of considering the system. As an analogy, in the electronics domain re-usable off-the-shelf electronic components have been available for many years in the form of integrated chips (ICs) which can be bought and deployed in other parts of the world. This is possible because each IC packages a clear set of functionality and provides a well-defined external interface.

In Robot System Architectures the interactions between software components are usually more complex compared to more traditional business applications. In Robotics, the software developer faces the complexity of event-based, reactive, and distributed interactions between sensors and motors and between several processing algorithms. Managing concurrent access to shared

resources by multiple (distributed) activities is one of the main issues, as thoroughly discussed in [12.85]. For this reason, robotic-specific component-based frameworks and toolkits have been developed, which offer mechanisms for real-time execution, synchronous and asynchronous communication, data flow and control flow management, and system configuration.

The following section reviews some of the most recent approaches and frameworks that have been specifically defined for robotics.

12.6.1 Agile Robot Development Network (aRDnet)

aRDnet [12.86] is a software suite developed at the German Aerospace Center (DLR) Institute of Robotics and Mechatronics that supports the development of distributed component-based systems for complex mechatronic systems with hard real-time requirements. The aRDnet suite has been used to implement computationally demanding control loops in the kHz range running over all the 41 degrees of freedom (DOF) of the upper humanoid body Justin, such as impedance control and collision avoidance. The robot control system is structured as a network of functional blocks and communication links distributed over a network of computers connected by a fast digital bus like the Gigabit-Ethernet.

An aRDnet functional block is a software module with multiple input and output ports that hide the actual transportation protocol, i. e., user datagram protocol (UDP) sockets or EtherCAT to exchange data with interconnected blocks. Blocks can be executed by individual processes or can be grouped in synchronization groups (e.g., for blocks interfacing the robot hardware) executed iteratively by a single process according to the *synchronous data flow* model of computation. Different synchronization groups can interact asynchronously through non blocking read and write operations.

12.6.2 Yet Another Robot Platform (YARP)

YARP [12.87] is an open-source project developed at the LIRA-Lab of the University of Genova in collaboration with MIT Computer Science and Artificial Intelligence Laboratory (CSAIL). It has been conceived with the same objectives as aRDnet, i. e., the development of distributed control systems for high DOF robots, such as the humanoid bodies. It consists in a lightweight software library for concurrent and distributed programming that has been used and tested on several operating systems.

Similarly to aRDnet, distributed software modules exchange data through input/output asynchronous communication ports. Differently from aRDnet, YARP ports can manage multiple connections for a given module at different data rates according to different protocols (e.g., transfer control protocol (TCP), UDP, multicast) allowing for the configuration of the quality of service (QOS) of the inter-module communication: an output port can send data to many destinations, while an input port can receive data from multiple connections.

12.6.3 Open Robot Control Software (OROCOS)

Orocos is one of the oldest open source framework in robotics, under development since 2001, and with professional industrial applications and products using it since about 2005 [12.88]. The focus of Orocos has always been to provide a hard real-time capable component framework, the so-called Real-Time Toolkit (RTT) implemented in C++ and as independent as possible from any communication middleware and operating system. Similarly to aRDnet and YARP, components interact with each other by exchanging data and events asynchronously through lock-free input/output ports according to the Data Flow communication paradigm.

The distinguish feature of OROCOS is the definition of a component model that specifies a standard behaviour for concurrent activities. Components with real-time, deterministic and cyclic behaviour get fixed and cyclic time budgets for computation and within a computation cycle they must reach stable intermediate states. OROCOS components are implemented as extension (inheritance) of the base class TaskContext, have their own thread of execution, and can be deployed as objects that share the same address space or as executables that communicate using the CORBA middleware.

12.6.4 Smartsoft

Smartsoft [12.89] is an open source framework that specifically addresses issues related to communication among software components of a robotic control system. Similarly to OROCOS, it defines a port-based component model, whose distinguished feature is a rich set of standard component interfaces (called communication patterns) with strictly defined interaction semantics:

- *Send*: Defines one-way communication with a client/server relationship
- *Query*: Two-way request communication with a client/server relationship

Avidbots Ex1016
Page 344 of 2154

- *Push newest/push timed*: 1-to-n distribution (broadcast) with a publisher/subscriber relationship
- *Event*: Asynchronous conditioned notification with a client/server relationship
- *Dynamic wiring*: Dynamic component wiring with a master/slave relationship.

Smartsoft is one of the first open source frameworks that has developed an open source toolchain for robotic software development based on the Eclipse Modeling Project. The toolchain implements a workflow that guides the software developer from high level design of the robot system architecture down to the components implementation and deployment through automatic model-to-model transformations.

12.6.5 Robot Operating System (ROS)

ROS [12.90] is a message-based peer-to-peer communication infrastructure supporting the easy integration of independently developed software components, called ROS nodes. A ROS system is thus a computation graph consisting of a set of nodes communicating with one another. Nodes are blocks of functional code and are implemented as classes that wrap robotic software libraries and provide access to the communication mechanisms of the underlying infrastructure (the ROS core). Nodes are organized into packages (file system folders containing libraries, nodes, and message definitions), which are grouped into thematic stacks (i. e., the navigation stack). Messages are typed data structures that can be nested into compound messages and are exchanged between nodes according to the publish/subscribe communication paradigm in an asynchronous manner without the need for the interacting nodes to know each other and to participate to the interaction at the same time. Messages are organized by topics, which correspond to information subjects that allow subscribers to recognize the events they are interested in. When a node receives a message belonging to a subscribed topic, a message handler is invoked asynchronously, which performs some computation on the message payload data and possibly generates a new message of a given topic to publish the computation results.

The distinguish characteristics of ROS is the lack of enforced architecture in ROS libraries, i. e., ROS nodes are designed as individually reusable components. An application is built by configuring individual nodes in such a way that they can exchange messages belonging to common topics. This characteristic favours the decentralized development of many small packages by experts in the various robotics sub-domains.

12.6.6 GenoM/BIP

GenoM [12.91] is a component-oriented software package developed by the LAAS CNRS robotics group that is used for specifying and implementing the functional level of robot system architectures. GenoM components are collections of control services, which execute a finite state automaton and are implemented by a set of C functions called codels, which get appropriately called during specific state transitions (i. e., start, exec, error, etc.). GenoM components exchange data and events through posters that are regions of shared memory.

The distinguish feature of the GenoM component model is the integration with the Behavior-Interaction-Priorities (BIPs) framework. BIP is a software framework and toolset for formally modeling and verifying complex, real-time component-based systems to guarantee correctness-by-construction of robotic control systems. BIP is used to produce a formal interaction model, which can be used for system level coordination to run (using the BIP engine) the functional layer composed of all the GenoM modules. BIP allows the hierarchical construction of compound components by using connectors, which interconnect the ports of GenoM components and models two basic modes of communication, namely, caller/provider and broadcaster/listener.

12.6.7 Best Practice in Robotics (BRICS)

Complementary to the projects described above BRICS [12.92], a joint research project funded by the European Commission, has formalized the robot development process according to the principles of Model Driven Engineering [12.93] and has provided tools, models, and functional libraries, which allow reducing the development time of robot software systems.

In particular, the concepts of Component Framework and Software Product Line have been introduced in the robot development process [12.94, 95] as illustrated in VIDEO 273. This approach promotes the routine use of existing software or software knowledge to construct new software, so that similarities in requirements, architectures and design between applications can be exploited to achieve substantial benefits in software quality, productivity, and business performance.

Avidbots Ex1016
Page 345 of 2154

12.7 Conclusions and Further Reading

Robot architectures are designed to facilitate the concurrent execution of task-achieving behaviors. They enable systems to control actuators, interpret sensors, plan, monitor execution, and deal with unexpected contingencies and opportunities. They provide the conceptual framework within which domain-dependent software development can take place, and they often provide programming tools that facilitate that development.

While no single architecture has proven to be best for all applications, researchers have developed a variety of approaches that can be applied in different situations. While there is not yet a specific formula for determining which architecture will be best suited for a given application, this chapter provides some guidelines to help developers in selecting the right architecture for the job. That being said, layered architectures have proven to be increasingly popular, due to their flexibility and ability to operate at multiple levels of abstraction simultaneously.

The book *Kortenkamp* et al. [12.96] provide several chapters on architectures that have influenced this chapter. Most text books in robotics [12.20, 97, 98] have sections on robot architectures. For many years in the mid 1990s, the AAAI Spring Symposia on Artificial Intelligence had sessions devoted to robot architectures, although proceedings from those symposia are not widely available. More information on GRACE can be found in [12.99–101].

Video-References

VIDEO 273 Software product line engineering for robotics
available from http://handbookofrobotics.org/view-chapter/12/videodetails/273

References

12.1 I. Jacobson, G. Booch, J. Rumbaugh: *The Unified Software Development Process* (Addison-Wesley, Reading 1998)

12.2 J.S. Albus: RCS: A reference model architecture for intelligent systems, Working Notes: AAAI 1995 Spring Symp. Lessons Learn. Implement. Software Archit. Phys. Agents (1995)

12.3 R.A. Brooks: A robust layered control system for a mobile robot, IEEE J. Robot. Autom. **2**(1), 14–23 (1986)

12.4 R.J. Firby: An investigation into reactive planning in complex domains, Proc. 5th Natl. Conf. Artif. Intel. (1987)

12.5 R. Simmons: Structured control for autonomous robots, IEEE Trans. Robot. Autom. **10**(1), 34–43 (1994)

12.6 J.J. Borrelly, E. Coste-Maniere, B. Espiau, K. Kapellos, R. Pissard-Gibollet, D. Simon, N. Turro: The ORCCAD architecture, Int. J. Robot. Res. **17**(4), 338–359 (1998)

12.7 B. Kuipers: The spatial semantic hierarchy, Artif. Intell. **119**, 191–233 (2000)

12.8 R. Orfali, D. Harkey: *Client/Server Programming with JAVA and CORBA* (Wiley, New York 1997)

12.9 R. Simmons, G. Whelan: Visualization tools for validating software of autonomous spacecraft, Proc. Int. Symp. Artif. Intel., Robot. Autom. in Space, Tokyo (1997)

12.10 R. A. Brooks: The Behavior Language: User's Guide, Tech. Rep. AIM-1227 (MIT, Artif. Intel. Lab, Cambridge 1990)

12.11 R.J. Firby, M.G. Slack: Task execution: Interfacing to reactive skill networks, Working Notes: AAAI Spring Symp. Lessons Learn. Implement. Archit. Phys. Agents, Stanford (1995)

12.12 E. Gat: ESL: A language for supporting robust plan execution in embedded autonomous agents, Proc. IEEE Aerosp. Conf. (1997)

12.13 V. Verma, T. Estlin, A. Jónsson, C. Pasareanu, R. Simmons, K. Tso: Plan execution interchange language (PLEXIL) for executable plans and command sequences, Proc. 8th Int. Symp. Artif. Intel. Robot. Autom. Space, Munich (2005)

12.14 S.A. Schneider, V.W. Chen, G. Pardo-Castellote, H.H. Wang: ControlShell: A software architecture for complex electromechanical systems, Int. J. Robot. Res. **17**(4), 360–380 (1998)

12.15 National Instruments: *LabVIEW* (National Instruments, Austin 2007) http://www.ni.com/labview/

12.16 N.J. Nilsson: A mobile automaton: an application of AI techniques, Proc. 1st Int. Joint Conf. Artif. Intel. (Morgan Kaufmann, San Francisco 1969) pp. 509–520

12.17 N.J. Nilsson: *Principles of Artificial Intelligence* (Tioga, Palo Alto 1980)

12.18 SRI International: Shakey the Robot, http://www.sri.com

12.19 P.E. Agre, D. Chapman: Pengi: An implementation of a theory of activity, Proc. 5th Natl. Conf. Artif. Intel. (1987)

12.20 R.C. Arkin: *Behavior-Based Robotics* (MIT Press, Cambridge 1998)

Avidbots Ex1016
Page 346 of 2154

12.21 J.H. Connell: SSS: A hybrid architecture applied to robot navigation, Proc. IEEE Int. Conf. Robot. Autom. (1992) pp. 2719–2724

12.22 M. Mataric: Integration of representation into goal-driven behavior-based robots, Proc. IEEE Int. Conf. Robot. Autom. (1992)

12.23 I. Horswill: Polly: A vision-based artificial agent, Proc. Natl. Conf. Artif. Intel. (AAAI) (1993)

12.24 D.W. Payton: An architecture for reflexive autonomous vehicle control, Proc. IEEE Int. Conf. Robot. Autom. (1986)

12.25 J.K. Rosenblatt: DAMN: A Distributed Architecture for Mobile Robot Navigation, Ph.D. Thesis (Carnegie Mellon Univ., Pittsburgh 1997)

12.26 R.C. Arkin: Motor schema-based mobile robot navigation, Int. J. Robot. Res. **8**(4), 92–112 (1989)

12.27 M. Arbib: Schema Theory. In: *Encyclopedia of Artificial Intelligence*, ed. by S. Shapiro (Wiley, New York 1992) pp. 1427–1443

12.28 O. Khatib: Real-time obstacle avoidance for manipulators and mobile robots, Proc. IEEE Int. Conf. Robot. Autom. (1985) pp. 500–505

12.29 R.C. Arkin: Integrating behavioral, perceptual, and world knowledge in reactive navigation, Robot. Autonom. Syst. **6**, 105–122 (1990)

12.30 R.C. Arkin, T. Balch: AuRA: Principles and practice in review, J. Exp. Theor. Artif. Intell. **9**(2/3), 175–188 (1997)

12.31 R.J. Firby: Adaptive Execution in Complex Dynamic Worlds, Ph.D. Thesis (Yale Univ., New Haven 1989)

12.32 R.J. Firby: Task networks for controlling continuous processes, Proc. 2nd Int. Conf. AI Plan. Syst. (1994)

12.33 R.P. Bonasso: Integrating reaction plans and layered competences through synchronous control, Proc. Int. Joint Conf. Artif. Intel. (1991)

12.34 S.J. Rosenschein, L.P. Kaelbling: The synthesis of digital machines with provable epistemic properties, Proc. Conf. Theor. Asp. Reas. Knowl. (1998)

12.35 L.P. Kaelbling: Goals as parallel program specifications, Proc. 6th Natl. Conf. Artif. Intel. (1988)

12.36 L. P. Kaelbling: *Compiling operator descriptions into reactive strategies using goal regression*, Tech. Rep., TR90-10, (Teleos Res., Palo Alto 1990)

12.37 R.P. Bonasso, R.J. Firby, E. Gat, D. Kortenkamp, D.P. Miller, M.G. Slack: Experiences with an architecture for intelligent, reactive agents, J. Exp. Theor. Artif. Intell. **9**(2/3), 237–256 (1997)

12.38 E. Gat: Integrating Planning and reacting in a heterogeneous asynchronous architecture for controlling real-world mobile robots, Proc. Natl. Conf. Artif. Intel. (AAAI) (1992)

12.39 G.N. Saridis: Architectures for intelligent controls. In: *Intelligent Control Systems: Theory and Applications*, ed. by S. Gupta (IEEE Press, Piscataway 1995)

12.40 R. Alami, R. Chatila, S. Fleury, M. Ghallab, F. Ingrand: An architecture for autonomy, Int. J. Robot. Res. **17**(4), 315–337 (1998)

12.41 M. Ghallab, H. Laruelle: Representation and control in IxTeT, a temporal planner, Proc. AIPS-94 (1994)

12.42 P. Laborie, M. Ghallab: Planning with sharable resource constraints, Proc. Int. Joint Conf. Artif. Intel. (1995)

12.43 M.P. Georgeff, F.F. Ingrand: Decision-making in an embedded reasoning system, Proc. Int. Joint Conf. Artif. Intel. (1989) pp. 972–978

12.44 F. Ingrand, R. Chatila, R. Alami, F. Robert: PRS: A high level supervision and control language for autonomous mobile robots, Proc. IEEE Int. Conf. Robot. Autom. (1996)

12.45 N.P. Muscettola, P. Nayak, B. Pell, B.C. Williams: Remote agent: To boldly go where no AI system has gone before, Artif. Intell. **103**(1), 5–47 (1998)

12.46 B.C. Williams, P.P. Nayak: A model-based approach to reactive self-configuring systems, Proc. AAAI (1996)

12.47 B. Sellner, F.W. Heger, L.M. Hiatt, R. Simmons, S. Singh: Coordinated multi-agent teams and sliding autonomy for large-scale assembly, Proc. IEEE **94**(7), 1425–1444 (2006), special issue on multi-agent systems

12.48 D. Goldberg, V. Cicirello, M.B. Dias, R. Simmons, S. Smith, A. Stentz: Market-based multi-robot planning in a distributed layered architecture, Proc. Int. Workshop Multi-Robot Syst (2003) pp. 27–38

12.49 J.S. Albus, R. Lumia, H.G. McCain: NASA/NBS Standard Reference model for Telerobot Control System Architecture (NASREM), Technol. #1235, (Nat. Inst. Stand, Gaithersburg 1986)

12.50 D.R. Blidberg, S.G. Chappell: Guidance and control architecture for the EAVE vehicle, IEEE J. Ocean Eng. **11**(4), 449–461 (1986)

12.51 J.S. Albus: Outline for a theory of intelligence, IEEE Trans. Syst. Man Cybern. **21**(3), 473–509 (1991)

12.52 R. Volpe, I. Nesnas, T. Estlin, D. Mutz, R. Petras, H. Das: The CLARAty architecture for robotic autonomy, Proc. IEEE Aerosp. Conf., Big Sky (2001)

12.53 I.A. Nesnas, R. Simmons, D. Gaines, C. Kunz, A. Diaz-Calderon, T. Estlin, R. Madison, J. Guineau, M. McHenry, I. Shu, D. Apfelbaum: CLARAty: Challenges and steps toward reusable robotic software, Int. J. Adv. Robot. Syst. **3**(1), 023–030 (2006)

12.54 T. Estlin, D. Gaines, C. Chouinard, F. Fisher, R. Castaño, M. Judd, R. Anderson, I. Nesnas: Enabling autonomous rover science through dynamic planning and scheduling, Proc. IEEE Aerosp. Conf., Big Sky (2005)

12.55 R. Knight, G. Rabideau, S. Chien, B. Engelhardt, R. Sherwood: CASPER: Space exploration through continuous planning, IEEE Intell. Syst. **16**(5), 70–75 (2001)

12.56 R. Simmons, D. Apfelbaum: A task description language for robot control, Proc. Conf. Intel. Robot Syst., Vancouver (1998)

12.57 T.A. Estlin, D. Gaines, C. Chouinard, R. Castaño, B. Bornstein, M. Judd, I.A.D. Nesnas, R. Anderson: Increased mars rover autonomy using AI planning, scheduling and execution, Proc. Int. Conf. Robot. Autom. (2007) pp. 4911–4918

Avidbots Ex1016
Page 347 of 2154

12.58 D. Musliner, E. Durfee, K. Shin: World modeling for dynamic construction of real-time control plans, Artif. Intell. **74**(1), 83–127 (1995)

12.59 D.J. Musliner, R.P. Goldman, M.J. Pelican: Using model checking to guarantee safety in automatically-synthesized real-time controllers, Proc. Int. Conf. Robot. Autom. (2000)

12.60 B. Espiau, K. Kapellos, M. Jourdan: Formal verification in robotics: Why and how?, Proc. Int. Symp. Robot. Res., Herrsching (1995)

12.61 G. Berry, G. Gonthier: The Esterel synchronous programming language: Design, semantics, implementation, Sci. Comput. Progr. **19**(2), 87–152 (1992)

12.62 M. Jourdan, F. Maraninchi, A. Olivero: Verifying quantitative real-time properties of synchronous programs, Lect. Notes Comput. Sci. **697**, 347–358 (1993)

12.63 G. Pardo-Castellote, S.A. Schneider: The network data delivery service: Real-time data connectivity for distributed control applications, Proc. Int. Conf. Robot. Autom. (1994) pp. 2870–2876

12.64 JAUS Reference Architecture Specification, Volume II, Part 1 Version 3.2: http://www.jauswg.org/baseline/refarch.html

12.65 JAUS Tutorial Powerpoint slides: http://www.jauswg.org/

12.66 JAUS Domain Model Volume I, Version 3.2: http://www.jauswg.org/baseline/current_baseline.shtml

12.67 E. Gat: ALFA: A language for programming reactive robotic control systems, Proc. IEEE Int. Conf. Robot. Autom. (1991) pp. 116–1121

12.68 R.A. Brooks: Elephants don't play chess, J. Robot. Autonom. Syst. **6**, 3–15 (1990)

12.69 L.P. Kaelbling: Rex – A symbolic language for the design and parallel implementation of embedded systems, Proc. 6th AIAA Comput. Aerosp. Conf., Wakefield (1987)

12.70 E. Gat: Non-linear sequencing and cognizant failure, Proc. AIP Conf. (1999)

12.71 E. Gat: On the role of stored internal state in the control of autonomous mobile robots, AI Mag. **14**(1), 64–73 (1993)

12.72 J.L. Peterson: *Petri Net Theory and the Modeling of Systems* (Prentice Hall, Upper Saddle River 1981)

12.73 K. Currie, A. Tate: O-Plan: The open planning architecture, Artif. Intell. **52**(1), 49–86 (1991)

12.74 D.S. Nau, Y. Cao, A. Lotem, H. Muñoz-Avila: SHOP: Simple hierarchical ordered planner, Proc. Int. Joint Conf. Artif. Intel. (1999) pp. 968–973

12.75 S. Chien, R. Knight, A. Stechert, R. Sherwood, G. Rabideau: Using iterative repair to improve the responsiveness of planning and scheduling, Proc. Int. Conf. AI Plan. Sched. (2000) pp. 300–307

12.76 N. Muscettola: HSTS: Integrating planning and scheduling. In: *Intelligent Scheduling*, ed. by M. Fox, M. Zweben (Morgan Kaufmann, San Francisco 1994)

12.77 R. Simmons, J. Fernandez, R. Goodwin, S. Koenig, J. O'Sullivan: Lessons learned from Xavier, IEEE Robot. Autom. Mag. **7**(2), 33–39 (2000)

12.78 R. Simmons: Inter Process Communication (Carnegie Mellon Univ., Pittsburgh 2007), http://www.cs.cmu.edu/IPC

12.79 S.W. Ambler: UML 2 Sequence Diagramms (Ambisoft, Toronto 2007) http://www.agilemodeling.com/artifacts/sequenceDiagram.htm

12.80 W.D. Smart: Writing code in the field: Implications for robot software development, Springer Tract. Adv. Robot. **30**, 93–105 (2007)

12.81 I.A.D. Nesnas: The CLARAty Project: Coping with hardware and software heterogeneity, Springer Tract. Adv. Robot. **30**, 31–70 (2007)

12.82 D. Brugali, E. Prassler: Software engineering for robotics, IEEE Robot. Autom. Mag. **16**(1), 9–15 (2009)

12.83 D. Brugali, P. Scandurra: Component-based robotic engineering (Part I), IEEE Robot. Autom. Mag. **16**(4), 84–96 (2009)

12.84 D. Brugali, A. Shakhimardanov: Component-based robotic engineering (Part II), IEEE Robot. Autom. Mag. **17**(1), 100–112 (2010)

12.85 D. Calisi, A. Censi, L. Iocchi, D. Nardi: Design choices for modular and flexible robotic software development: The OpenRDK viewpoint, J. Software Eng. Robot. **3**(1), 13–27 (2012)

12.86 B. Baeuml, G. Hirzinger: When hard realtime matters: Software for complex mechatronic systems, Robot. Auton. Syst. **56**, 5–13 (2008)

12.87 G. Metta, P. Fitzpatrick, L. Natale: YARP: Yet another robot platform, Int. J. Adv. Robot. Syst **3**(1), 43–48 (2006)

12.88 H. Bruyninckx: Open robot control software: the OROCOS project, Proc. IEEE Int. Conf. Robot. Autom. (2001) pp. 2523–2528

12.89 C. Schlegel, A. Steck, D. Brugali, A. Knoll: Design abstraction and processes in robotics: From code-driven to model-driven engineering, Lect. Notes Comput. Sci. **6472**, 324–335 (2010)

12.90 S. Cousins: Welcome to ROS Topics, IEEE Robot. Autom. Mag. **17**(1), 13–14 (2010)

12.91 S. Bensalem, L. de Silva, F. Ingrand, R. Yan: A verifiable and correct-by-construction controller for robot functional levels, J. Software Eng. Robot. **2**(1), 1–19 (2011)

12.92 R. Bischoff, T. Guhl, E. Prassler, W. Nowak, G. Kraetzschmar, H. Bruyninckx, P. Soetens, M. Haegele, A. Pott, P. Breedveld, J. Broenink, D. Brugali, N. Tomatis: BRICS: Best practice in robotics, Proc. 41st Int. Symp. Robot., Munich (2010) pp. 7–9

12.93 M. Klotzbuecher, N. Hochgeschwender, L. Gherardi, H. Bruyninckx, G. Kraetzschmar, D. Brugali: The BRICS component model: A model-based development paradigm for complex robotics software systems, 28th ACM Symp. Appl. Comput., Coimbra (2013) pp. 18–22

12.94 D. Brugali, L. Gherardi, A. Luzzana, A. Zakharov: A reuse-oriented development process for component-based robotic systems, Lect. Notes Comput. Sci. **7628**, 361–374 (2012)

12.95 L. Gherardi, D. Brugali: Modeling and reusing robotic software architectures: The hyperflex

Avidbots Ex1016
Page 348 of 2154

toolchain, Proc. IEEE Robot. Autom. Conf. ICRA '14, Hong Kong (2014)
12.96 D. Kortenkamp, R.P. Bonasso, R. Murphy: *Artificial Intelligence and Mobile Robots* (MIT Press, Cambridge 1998)
12.97 R. Murphy: *Introduction to AI Robotics* (MIT Press, Cambridge 2000)
12.98 R. Siegwart, I.R. Nourbakhsh: *Introduction to Autonomous Mobile Robots* (MIT Press, Cambridge 2004)
12.99 R. Simmons, D. Goldberg, A. Goode, M. Montemerlo, N. Roy, B. Sellner, C. Urmson, A. Schultz, M. Abramson, W. Adams, A. Atrash, M. Bugajska, M. Coblenz, M. MacMahon, D. Perzanowski, I. Horswill, R. Zubek, D. Kortenkamp, B. Wolfe, T. Milam, B. Maxwell: GRACE: An autonomous robot for the AAAI Robot Challenge, AAAI Mag. **24**(2), 51–72 (2003)
12.100 R. Gockley, R. Simmons, J. Wang, D. Busquets, C. DiSalvo, K. Caffrey, S. Rosenthal, J. Mink, S. Thomas, W. Adams, T. Lauducci, M. Bugajska, D. Perzanowski, A. Schultz: Grace and George: Social Robots at AAAI, Proc. AAAI Mob. Robot Comp. Workshop (2004), pp. 15–20
12.101 M.P. Michalowski, S. Sabanovic, C. DiSalvo, D. Busquets, L.M. Hiatt, N.A. Melchior, R. Simmons: Socially Distributed Perception: GRACE plays social tag at AAAI 2005, Auton. Robot. **22**(4), 385–397 (2007)

Avidbots Ex1016
Page 349 of 2154

Multimedia Contents



13. Behavior-Based Systems

François Michaud, Monica Nicolescu

Nature is filled with examples of autonomous creatures capable of dealing with the diversity, unpredictability, and rapidly changing conditions of the real world. Such creatures must make decisions and take actions based on incomplete perception, time constraints, limited knowledge about the world, cognition, reasoning and physical capabilities, in uncontrolled conditions and with very limited cues about the intent of others. Consequently, one way of evaluating intelligence is based on the creature's ability to make the most of what it has available to handle the complexities of the real world. The main objective of this chapter is to explain *behavior-based systems* and their use in autonomous control problems and applications. The chapter is organized as follows. Section 13.1 overviews robot control, introducing behavior-based systems in relation to other established approaches to robot control. Section 13.2 follows by outlining the basic principles of behavior-based systems that make them distinct from other types of robot control architectures. The concept of basis behaviors, the means of modularizing behavior-based systems, is presented in Sect. 13.3. Section 13.4 describes how behaviors are used as building blocks for creating representations for use by behavior-based systems, enabling the robot to reason about the world and about itself in that world. Section 13.5 presents several different classes of learning methods for behavior-based systems, validated on single-robot and multi-robot systems. Section 13.6 provides an overview of various robotics problems and application domains that have successfully been addressed or are currently being studied with behavior-based control. Finally, Sect. 13.7 concludes the chapter.

13.1 **Robot Control Approaches** 308
13.1.1 Deliberative – Think, Then Act 308
13.1.2 Reactive – Don't Think, (Re)Act ... 308
13.1.3 Hybrid – Think and Act Concurrently 309
13.1.4 Behavior-Based Control – Think the Way You Act 309
13.1.5 When to Use What 309
13.2 **Basic Principles of Behavior-Based Systems** 310
13.2.1 Misconceptions 312
13.3 **Basis Behaviors** 313
13.4 **Representation in Behavior-Based Systems** 313
13.5 **Learning in Behavior-Based Systems** 314
13.5.1 Reinforcement Learning in Behavior-Based Systems 315
13.5.2 Learning Behavior Networks 315
13.5.3 Learning from Demonstration in Behavior-Based Systems 316
13.5.4 Learning from History of Behavior Use 317
13.6 **Applications and Continuing Work** 318
13.6.1 Adaptive Behavior Selection – A Case Study 319
13.7 **Conclusions and Further Reading** 322
Video-References 322
References 323

Avidbots Ex1016
Page 350 of 2154

13.1 Robot Control Approaches

Situated robotics deals with embodied machines in complex, challenging, often dynamically changing environments. *Situatedness* thus refers to existing in a complex, challenging environment, and having one's behavior strongly affected by it. In contrast, robots that exist in static, unchanging environments are usually not thought to be situated. These include assembly robots operating in complex but highly structured, fixed, and strongly predictable environments, specifically engineered and controlled to enable the robot to accomplish very specific tasks. The predictability and stability of the environment have a direct impact on the complexity of the robot that must operate in it; situated robots therefore present a significant challenge for the designer.

Robot control, also referred to as robot decision-making or robot computational architecture, is the process of taking information about the environment through the robot's sensors, processing it as necessary in order to make decisions about how to act, and executing actions in the environment. The complexity of the environment, i. e., the level of situatedness, has a direct impact on the complexity of control, which is, in turn, directly related to the robot's task. Control architectures are covered in Chap. 12 of the Handbook.

While there are infinitely many possible ways to program a robot, there are fundamentally four classes of robot control methods, described below.

13.1.1 Deliberative – Think, Then Act

In deliberative control, the robot uses all of the available sensory information, and all of the internally stored knowledge, to reason about what actions to take next. The control system is usually organized using a functional decomposition of the decision-making processes, consisting of a sensory processing module, a modeling module, a planning module, a value judgment module, and an execution module [13.1]. Such functional decomposition allows complex operations to be performed, but implies strong sequential interdependencies between the decision-making modules.

Reasoning in deliberative systems is typically in the form of planning, requiring a search of possible state–action sequences and their outcomes. *Planning*, a major component of artificial intelligence, is known to be a computationally complex process. The process requires the robot to perform a sequence of sense–plan–act steps (e.g., *combine the sensory data into a map of the world, then use the planner to find a path in the map, then send steps of the plan to the robot's wheels*) [13.2–4]. The robot must construct and then potentially evaluate all possible plans until it finds one that enables it to reach its goal, solve the task, or decide on a trajectory to execute. Shakey, an early mobile robot that used STRIPS, a general planner, is an example of such a system applied to the problem of avoiding obstacles and navigating based on vision data [13.5].

Planning requires the existence of an internal, symbolic representation of the world, which allows the robot to look ahead into the future and predict the outcomes of possible actions in various states, so as to generate plans. The internal model, thus, must be kept accurate and up to date. When there is sufficient time to generate a plan and the world model is accurate, this approach allows the robot to act strategically, selecting the best course of action for a given situation. However, being situated in a noisy, dynamic world usually makes this impossible [13.6, 7]. Today, no situated robots are purely deliberative. The advent of alternative architectures was driven by the need for faster yet appropriate action in response to the demands of complex and dynamically changing real-world environments.

13.1.2 Reactive – Don't Think, (Re)Act

Reactive control is a technique for tightly coupling sensory inputs and effector outputs, typically involving no intervening reasoning [13.8] to allow the robot to respond very quickly to changing and unstructured environments [13.9]. Reactive control is inspired by the biological notion of *stimulus–response*; it does not require the acquisition or maintenance of world models, as it does not rely on the types of complex reasoning processes utilized in deliberative control. Rather, rule-based methods involving a minimal amount of computation, and no internal representations or knowledge of the world are typically used. Reactive systems achieve rapid real-time responses by embedding the robot's controller in a collection of preprogrammed, concurrent condition–action rules with minimal internal state (e.g., *if bumped*, *stop*; *if stopped*, *back up*) [13.8, 10]. This makes reactive control especially well suited to dynamic and unstructured worlds where having access to a world model is not a realistic option. Furthermore, the minimal amount of computation involved means that reactive systems are able to respond in a timely manner to rapidly changing environments.

Reactive control is a powerful and effective control method that abounds in nature; insects, which vastly outnumber vertebrates, are largely reactive. However, limitations to pure reactivity include the inability to store (much if any) information or have memory or internal representations of the world [13.11], and therefore the inability to learn and improve over time.

Avidbots Ex1016
Page 351 of 2154

Reactive control trades off complexity of reasoning for fast reaction time. Formal analysis has shown that, for environments and tasks that can be characterized a priori, reactive controllers can be very powerful, and if properly structured, capable of optimal performance in particular classes of problems [13.12, 13]. In other types of environments and tasks where internal models, memory, and learning are required, reactive control is not sufficient.

13.1.3 Hybrid – Think and Act Concurrently

Hybrid control aims to combine the best aspects of reactive and deliberative control: the real-time response of reactivity and the rationality and optimality of deliberation. As a result, hybrid control systems contain two different components, the reactive/concurrent condition–action rules and the deliberative ones, which must interact in order to produce a coherent output. This is challenging because the reactive component deals with the robot's immediate needs, such as moving while avoiding obstacles, and thus operates on a very fast time scale and uses direct external sensory data and signals. In contrast, the deliberative component uses highly abstracted, symbolic, internal representations of the world, and operates on them on a longer time scale, for example to perform global path planning or plan for high-level decision-making. As long as the outputs of the two components are not in conflict, the system requires no further coordination. However, the two parts of the system must interact if they are to benefit from each other. Consequently, the reactive system must override the deliberative one if the world presents some unexpected and immediate challenge. Analogously, the deliberative component must inform the reactive one in order to guide the robot toward more efficient and optimal trajectories and goals. The interaction of the two parts of the system requires an intermediate component, which reconciles the different representations used by the other two and any conflicts between their outputs. The construction of this intermediate component is typically the greatest challenge of hybrid system design.

Hybrid systems are referred to as layered, or tiered, robot control architecture (Chap. 12). Many are referred to as *three-layer architectures* because of their structure, which consists of the reactive (execution) layer, intermediate (coordination) layer, and deliberative (organization/planning) layer. The layers are organized according to the principle of increasing precision of control in the lower layers with decreasing intelligence [13.14]. A great deal of research has been invested into the design these components and their interactions [13.2, 15–21].

Three-layer architectures aim to harness the best of reactive control in the form of dynamic, concurrent and time-responsive control, and the best of deliberative control, in the form of globally efficient actions over a long time scale. However, there are complex issues involved in interfacing these fundamentally differing components, and the manner in which their functionality should be partitioned is not yet well understood [13.22].

13.1.4 Behavior-Based Control – Think the Way You Act

Behavior-based control employs a set of distributed, interacting modules, called *behaviors*, that collectively achieve the desired system-level behavior. To an external observer, behaviors are patterns of the robot's activity emerging from interactions between the robot and its environment. To a programmer, behaviors are control modules that cluster sets of constraints in order to achieve and maintain a goal [13.22, 23]. Each behavior receives inputs from sensors and/or other behaviors in the system, and provides outputs to the robot's actuators or to other behaviors. Thus, a behavior-based controller is a structured network of interacting behaviors, with no centralized world representation or focus of control. Instead, individual behaviors and networks of behaviors maintain any state information and models.

Well-designed behavior-based systems take advantage of the dynamics of interaction among the behaviors themselves, and between the behaviors and the environment. The functionality of behavior-based systems can be said to emerge from those interactions and is thus neither a property of the robot or the environment in isolation, but rather a result of the interplay between them [13.22]. Unlike reactive control, which utilizes collections of reactive rules with little if any state and no representation, behavior-based control utilizes collections of behaviors, which have no such constraints: behaviors do have state and can be used to construct representations, thereby enabling reasoning, planning, and learning.

13.1.5 When to Use What

Characterizing a given robot computational architecture based on these four classes of control is often a matter of degree, as architectures attempt to combine the advantages of these paradigms, especially the responsiveness, robustness, and flexibility of the behavior-based approach with the use of abstract representational knowledge for reasoning and planning about the world [13.22] or for managing multiple

conflicting goals. For example, AuRA (autonomous robot architecture) uses a planner to select behaviors [13.22] and 3T uses behaviors in the execution layer of a three-level hierarchical architecture [13.24]; both of these architectures dynamically reconfigure behaviors according to reasoning based on available world knowledge [13.22]. *Goller* et al. [13.25] use a set of 15 navigational behaviors with 17 activating behaviors, combined with a task planner and a topological navigation module.

Each of the above approaches to robot control has its strengths and weaknesses, and all play important and successful roles in certain robot control problems and applications. Each offers interesting but different insights, and no single approach should be seen as ideal or otherwise in the absolute. Rather, the choice of robot control methodology should be based on the particular task, environment, and robot. Robot control presents fundamental tradeoffs having to do with time scale of response, system organization, and modularity: thinking allows looking ahead to avoid mistakes, but only as long as sufficient, accurate, up-to-date information is available, otherwise reacting may be the best way to handle the world. As a consequence of these inherent tradeoffs, it is important to have different methodologies at our disposal rather than having to fit all controller needs into a single methodology. Selecting an appropriate control methodology and designing an architecture within it is best determined by the situatedness properties of the problem, the nature of the task, the level of efficiency or optimality needed, and the capabilities of the robot, both in terms of hardware, world modeling, and computation.

For example, reactive control is the best choice for environments demanding immediate response, but such speed of reaction comes at the price of being myopic, not looking into the past or the future. Reactive systems are also a popular choice in highly stochastic environments, and environments that can be properly characterized so as to be encoded in a reactive input–output mapping. Deliberative systems, on the other hand, are the only choice for domains that require a great deal of strategy and optimization, and in turn search and planning. Such domains, however, are not typical of situated robotics, but more so of scheduling, game playing, and system configuration, among others. Hybrid systems are well suited for environments and tasks where internal models and planning are needed, and the real-time demands are few, or sufficiently independent of the higher-level reasoning.

Behavior-based systems, in contrast, are best suited for environments with significant dynamic changes, where fast response and adaptivity are crucial, but the ability to do some looking ahead and avoid past mistakes is required. Those capabilities are spread over the active behaviors, using representations if necessary [13.23], as discussed in the following.

13.2 Basic Principles of Behavior-Based Systems

Behavior-based robotics was developed for situated robots, allowing them to adapt to the dynamics of real-world environments without operating upon abstract representations of reality [13.11], but also giving them more computational capability and expressivity than are available to reactive robots. Behavior-based systems maintain a tight coupling of sensing and action through behaviors, and use the behavior structure for representation and learning. Therefore, it is uncommon for a behavior to perform extensive computation or reasoning relying on a traditional world model, unless such computation can be done in a timely manner in response to dynamic and fast-changing environment and task demands.

Figure 13.1 summarizes the general components of low-level behavior-based systems. Note that there is a distinction between *activation conditions*, which allow the behavior to generate actions, and *stimuli*, from which actions are generated.

The basic principles of behavior-based control can be summarized briefly as follows:

- Behaviors are implemented as control laws (sometimes similar to those used in control theory), either in software or hardware, as a processing element or as a procedure.
- Each behavior can take inputs from the robot's sensors (e.g., proximity sensors, range detectors, contact sensors, camera) and/or from other modules in the system, and can send outputs to the robot's effectors (e.g., wheels, grippers, arm, speech) and/or to other modules.
- Many different behaviors may independently receive input from the same sensors and output action commands to the same actuators.
- Behaviors are encoded to be relatively simple, and are added to the system incrementally.
- Behaviors (or subsets thereof) are executed concurrently, not sequentially, in order to exploit par-

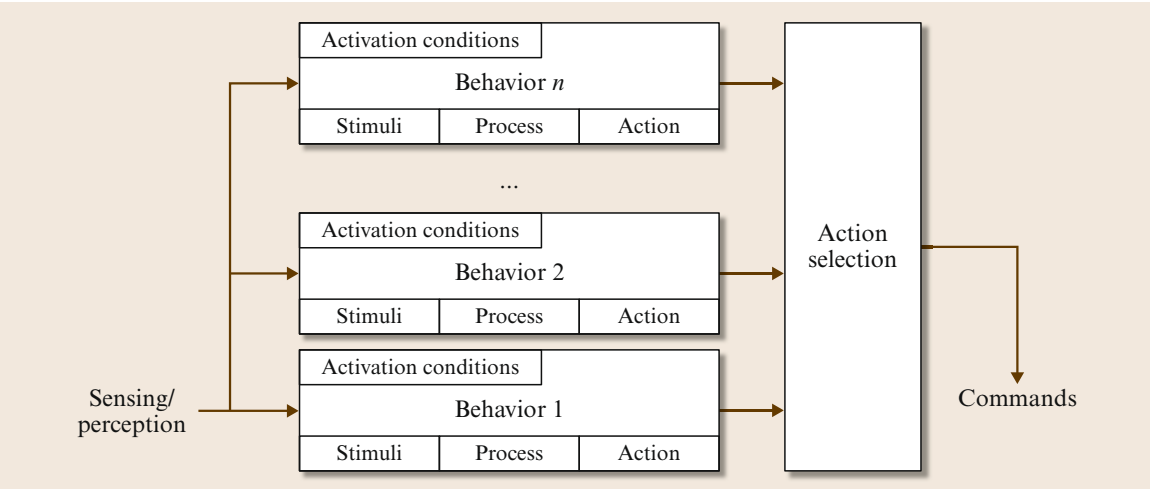


Fig. 13.1 A general schematic of one type of behavior-based systems

allelism and speed of computation, as well as the interaction dynamics among behaviors and between behaviors and the environment.

Behaviors are designed at a variety of abstraction levels, facilitating bottom-up construction of behavior-based systems. New behaviors are introduced into the system incrementally, from the simple to the more complex, until their interaction results in the desired overall capabilities of the robot. In general, behaviors encode time-extended processes, not atomic actions that are typical of feedback control (e.g., *go-forward-by-a-small-increment* or *turn-by-a-small-angle*). The interaction and integration of temporal and spatial effects are of key importance in behavior-based systems. Merely having one process controlling an actuator for predetermined intervals of time, or using as many processes as there are effectors to control them, does not suffice as the basis for behavior-based control. It is the combined effect of concurrent processes over time and driven by perception and internal states that creates the relevant behavior-based dynamics in a control system.

As a first step, survival behaviors, such as *collision-avoidance*, are implemented. These behaviors are often reactive in nature, since reactive rules can and often do form components of simple behaviors. Next, behaviors are added that provide more complex capabilities, such as *wall-following*, *target-chasing*, *homing*, *find-object*, *get-recharged*, *avoid-the-light*, *aggregate-with-group*, *pick-up-object*, *find-landmark*. Depending on the system being designed, behaviors implementing distributed representations may be added, as may be behaviors capable of learning about the world and/or the robot itself, and operating on those representations and learned information. Representation and learning are addressed in more detail in Sect. 13.4.

Behavior-based systems are typically designed so the effects of the behaviors interact largely in the environment rather than internally through the system, taking advantage of the richness of the interaction dynamics by exploiting the properties of situatedness. These dynamics are sometimes called *emergent* behaviors because they *emerge* from the interactions and are not internally specified by the robot's program. Therefore, the internal behavior structure of a behavior-based system need not necessarily mirror its externally manifested behavior. For example, a robot that flocks with other robots may not have a specific *flocking* behavior; instead, its interaction with the environment and other robots may result in flocking, although its only behaviors may be *avoid-collisions*, *stay-close-to-the-group*, and *keep-going*.

For such an approach to work, a behavior-based system must resolve the issue of choosing a particular action or behavior from multiple options, a process known as *action selection* [13.26] (or also *behavior coordination* [13.27], behavior selection [13.28], or behavior fusion [13.29]) problem. This is one of the central design challenges of behavior-based systems. One approach to action selection is the use of a predefined behavior hierarchy, in which commands from the highest-ranking active behavior are sent to the actuator and all others are ignored. Numerous approaches based on other principles as well as ad hoc methods for addressing the action selection problem have been developed and demonstrated on robotic systems. These methods aim to provide increased flexibility but, in some cases, may do so at the cost of reducing the ef-

Avidbots Ex1016
Page 354 of 2154

ficiency or the analyzability of the resulting control systems. Developed methods include varieties of motor schemas [13.16], command fusion [13.30], spreading of activation through a behavior network [13.31, 32], behavior auctions [13.33, 34], and fuzzy logic [13.35, 36], among many others. *Pirjanian* [13.27] presents a survey of action selection mechanisms. Some frameworks also support the use of several action selection mechanisms, such as APOC [13.28] (allowing dynamic selection and changes) and iB2C [13.29] (along with development guidelines, analysis tools and visualization techniques).

13.2.1 Misconceptions

Because behavior-based systems are not always simple to describe or implement, they are also often misunderstood. The most common misconception equates reactive and behavior-based systems. Historically, the advent of behavior-based systems was inspired by reactive systems; both maintain real-time couplings between sensing and action [13.18, 37], and are structured and developed bottom-up, consisting of distributed modules. However, behavior-based systems are fundamentally *more powerful* than reactive systems, because they can store representations [13.38], while reactive systems cannot. Reactive systems are limited by their lack of internal state; they are incapable of using internal representations and learning. Behavior-based systems overcome this limitation because their underlying unit of representation, the *behavior*, can store state internally, in a distributed fashion.

The means by which state and representation are distributed in behavior-based systems is one of the sources of the flexibility of the control methodology. Representations in behavior-based systems are distributed, so as to best match and utilize the underlying behavior structure that causes the robot to act. This is how *thinking* can be organized in much the same way as *acting*. Thus if a robot needs to plan ahead, it does so in a network of communicating behaviors, rather than a single centralized planner. If a robot needs to store a large map, the map might be distributed over multiple behavior modules representing its components, such as a network of landmarks, as in [13.39], or a network of parameterized navigation behaviors, as in [13.40, 41], so that reasoning about the map/environment/task can be done in an active fashion, through using message passing within the behavior network. The planning and reasoning components of the behavior-based system use the same mechanisms as the sensing- and action-oriented behaviors, and as a result do not operate on a fundamentally different time scale and representation relative to one another. Various forms of distributed representations are used, ranging from static table structures and networks, to active procedural processes implemented within the behavior networks.

Another area of common misconception relates to the comparison between behavior-based systems and hybrid systems. Because the two use such different modularization strategies, it is often assumed that one approach (usually hybrid) has improved expressive capabilities. In fact, behavior-based and hybrid systems have the same expressive and computational capabilities: both can exploit representations and look ahead, but they do so in very different ways. This has resulted in different application domains being best suited to behavior-based versus hybrid systems. Specifically, hybrid systems dominate the domain of single-robot control, unless the task is so time-demanding that a reactive system must be used. Behavior-based systems dominate the domain of multi-robot control because the notion of collections of behaviors within the system scales well to collections of such robots, resulting in robust, adaptive group behavior [13.42, 43]. See Chap. 53 on multiple mobile robot systems for more details.

Like hybrid systems, behavior-based systems may be organized in layers, but unlike hybrid systems, the layers do not differ from each other drastically in terms of organization, time scale or representation used. Behavior-based systems typically do not employ the hierarchical/sequential division favored by hybrid approaches. Behavior-based systems do provide both low-level control and high-level deliberation; the latter can be performed by one or more distributed representations that compute(s) over the other behaviors or modules, often directly utilizing low-level behaviors and their outputs. The resulting systems, built from the bottom-up, do not divide into differently represented and independent components, and consist of elements directly tied in some ways to behaviors. The power, elegance, and complexity of behavior-based systems all stem from the ways in which their constituent behaviors are designed, coordinated, and used. For instance, monitoring how behaviors for low-level control contribute to the commands issued over time can be used to evaluate how appropriate the robot operates in its environment based on its set of activated behaviors. Different methods can be used to implement such a mechanism, such as temporal analyses [13.36, 44–47], pattern recognition techniques [13.48] and graph-based representations [13.49–51].

Therefore, the principles to keep in mind when extending behavior-based to higher decisional levels are:

Avidbots Ex1016
Page 355 of 2154

1. Use behaviors as the building block of both decision-making and action execution processes;
2. Use distributed parallel evaluation and concurrent control over lower-level behaviors, which take real-time inputs from sensory data and send real-time commands to effectors;
3. Have no centralized components, each module carrying out its own responsibilities.

The following sections describe and illustrate in more detail how behavior-based principles can be used to control robots.

13.3 Basis Behaviors

The process of designing a set of behaviors for a robot is referred to as *behavior synthesis*, and is typically performed by hand, although some methods for automated synthesis behaviors have been developed and successfully demonstrated [13.52, 53]. In all cases, behaviors perform a particular activity, attain a goal, or maintain some state. The notion of defining an *optimal* behavior set for a given robot or task has been considered, but it is generally accepted that such a notion is not realistic as it is dependent on too many specifics of a given system and environment that cannot currently be effectively formalized.

Matarić et al. [13.43, 54] describe *basis behaviors*, also referred to as *primitive behaviors*, as a tool for structuring and thus simplifying behavior synthesis. Basis behaviors are a set of behaviors such that each is *necessary*, in the sense that each either achieves, or helps to achieve, a relevant goal that cannot be achieved without it by other members of that set. Furthermore, the basis behavior set is *sufficient* for achieving the goals mandated for the controller. The term *basis* was chosen to be indicative of the similar notion within linear algebra. The property of necessity or parsimony is analogous to the idea of linear independence; the idea of sufficiency is similar to the linear algebraic concept of span. Basis behaviors should be simple, stable, robust, and scalable.

Another organizational principle of basis behaviors is *orthogonality*. Two behaviors are orthogonal if they do not interfere with one another, each inducing no side-effects in the other. This is often achieved by having behaviors take mutually exclusive sensory inputs. Another method is to have different behaviors control separate effectors. This form of factorization is only feasible when the robot's dynamics do not inhibit their separability. In contrast, autonomous helicopter control is an example of a highly coupled system; *Saripalli* et al. [13.55] demonstrated how behavior-based control can be effectively applied to robust autonomous helicopter flight.

Basis behavior design principles have been applied to single-robot as well as multi-robot behavior-based systems in a variety of applications, ranging from navigation to foraging, coordinated group movement, box pushing, and others.

13.4 Representation in Behavior-Based Systems

Embedding representation into behavior-based systems involves the challenge of conserving the basic principles of the approach at all levels of system decision-making. Combining abstract reasoning processes with behaviors must be done in a way that exploits interaction dynamics and desirable emergent system properties.

Matarić et al. [13.38, 56] describe work with a robot named *Toto* and as shown by VIDEO 35, which introduced the use of distributed representation into behavior-based systems. Toto's capabilities included safe navigation, landmark detection, map learning, and path planning in the learned map representation, all within the behavior-based framework. To exploit the principles underlying behavior-based control, Toto's representation was not a centralized map. Instead, any newly discovered landmark in the environment was assigned to a new map representation behavior, which stored the landmark descriptor (type, estimated Cartesian location, and compass orientation). Whenever sensory inputs matched the landmark descriptor, the robot localized to that particular landmark and the behavior became active. The following is pseudo-code for each landmark behavior:

Algorithm 13.1

```
my-behavior-type: C
my-compass-direction: 0
my-approximate-location: (x,y)
my-approximate-length: 6.5
whenever received (input)
```

```
if input(behavior-type) = my-behavior-type
   and
      input(compass-direction) =
      my-compass-direction then
   active ¡- true
end if
```

As new landmarks are discovered, they are added to the map representation behavior network. In this way, the topology of the resulting map network is isomorphic to the topology of the network graph in the physical environment Toto has explored. The edges in the network graph are also communication links in the behavior network, allowing landmark behaviors to communicate through local message passing. Consequently, a currently active map behavior can send a message to its topological neighbor(s), thereby indicating expectation of it being the next recognized landmark and facilitating Toto's localization. Planning in the network takes place through the use of the same message-passing mechanism. The goal landmark (which could be selected by the user as part of the task, such as *go to this particular corridor* or *go to the nearest north-facing wall*), sends messages (i. e., spreads activation) from itself to its neighbors, which pass it on throughout the network. As messages are passed, the length of each landmark in the graph is accrued, thereby estimating the length of each path. The shortest path arriving at the currently active network behavior indicates the best direction to pursue toward the goal. This is equivalent to a distributed Dijkstra search. Importantly, because this search is an active ongoing process within a behavior map rather than a static process (as it would be in a centralized map representation), if the robot is picked up and moved to another location, as soon as it localizes, it continues on the optimal path to the goal: each landmark makes a local decision as to where to go next toward the goal, and no unique global path is stored in any central location/representation. Thus, the path is constantly refreshed and updated; if any route is blocked, the link in the graph is disconnected and the shortest path is updated dynamically.

Toto exemplifies how, in a behavior-based system, a representation can be stored in a distributed fashion, so as to best match the underlying behavior structure that produces the robot's external goal-driven activity. If Toto needs to make high-level decisions (such as planning ahead to a distant goal), it does so in a network of communicating behaviors, rather than a single centralized component. This results in scalable and efficient computation for the system as a whole, since the typically slower decision-making processes such as planning are distributed and modularized in a way that makes them more consistent with the time scale and representation of the rest of the system. Note the fundamental difference between this general attempt of behavior-based systems to homogenize their representation through the use of behaviors as universal modules, compared to hybrid systems which rely on inherently different representations and time scales at different levels of the system.

13.5 Learning in Behavior-Based Systems

The ability to improve performance over time and to reason about the world, in the context of a changing and dynamic environment, are important areas of research in situated robotics. Unlike in classical machine learning where the goal is typically to optimize performance over a long period of time, in situated learning the aim is to adapt relatively quickly toward attaining efficiency in the light of uncertainty. Models from biology are often considered, given its properties of learning directly from environmental feedback. Variations and adaptations of machine learning, and in particular reinforcement learning, have been effectively applied to behavior-based robots, which have demonstrated learning to walk [13.57], communicate [13.58], navigate and create topological maps [13.38, 59], divide tasks [13.23, 60], behave socially [13.61], and even identify opponents and score goals in robot soccer [13.62]. Methods from artificial life, evolutionary computation/genetic algorithms, fuzzy logic, vision and learning, multi-agent systems, and many other research areas continue to be actively explored and applied to behavior-based robots as their role in animal modeling and practical applications continue to develop.

When operating in unpredictable and partially observable environments, an autonomous robot must examine the evolution of its general states, and try to capture what emerges from the interaction dynamics with its environment. Temporal integration of different types of observations is an important aspect of that capability [13.63, 64]. Work on motivational systems [13.47, 65–68] has shown that a balance between planning and reactivity for goal management can be achieved using internal variables activated or inhibited by different factors [13.42, 46, 69, 70]. Motivations can be cyclic (e.g., circadian rhythms) or change in various temporally dependent ways [13.68]. In general, the

notion of motivations is used to efficiently balance the need to adapt to the contingencies of the world and to accomplish the robot's goals.

In the following subsections, we discuss four successfully validated classes of learning approaches in behavior-based systems. The approaches differ in what is learned and where learning algorithms are applied, but in all cases behaviors are used as the underlying building blocks for the learning process.

13.5.1 Reinforcement Learning in Behavior-Based Systems

Behaviors are recognized as excellent substrates for speeding up reinforcement learning (RL), which is known to suffer from the curse of dimensionality. The earliest examples of RL in behavior-based systems demonstrated hexapod walking [13.57] and box-pushing [13.71]. Both decomposed the control system into a small set of behaviors, and used generalized input states, thereby effectively reducing the size of the state space. In the box-pushing example, the learning problem was broken up into modularized policies for learning separate mutually exclusive behaviors: for getting out when stuck, for finding the box when lost and not stuck, and for pushing the box when in contact with one and not stuck. The modularization into behaviors resulted in greatly accelerated as well as more robust learning. More recently, *Kober* et al. [13.72] demonstrated the use of reinforcement learning to learn a set of meta-parameters that enable the adaptation of learned motor plans, such as a table tennis pass or a dart throwing task, to new situations.

Scaling up reinforcement learning to multi-robot behavior-based systems has also been explored [13.23, 73]. In multi-robot systems, the environment presents further challenges of nonstationarity and credit assignment, due to the presence of other agents and concurrent learners. The problem was studied in the context of a foraging task with four robots, each initially equipped with a small set of basis behaviors (*searching*, *homing*, *picking-up*, *dropping*, *following*, *avoiding*) and learning individual behavior selection policies, i. e., which behavior to execute under which conditions. Due to interference among concurrent learners, this problem could not be solved directly by standard RL. *Shaping*, a concept from psychology [13.74], was introduced; it was subsequently adopted in robot RL [13.75]. Shaping pushes the reward closer to the subgoals of the behavior, and thus encourages the learner to incrementally improve its behaviors by searching the behavior space more effectively. *Matarić* [13.73] introduced shaping through *progress estimators*, measures of progress toward the goal of a given behavior during its execution. This form of reward shaping addresses two issues associated with delayed reward: behavior termination and fortuitous reward. Behavior termination is event-driven; the duration of any given behavior is determined by the interaction dynamics with the environment, and can vary greatly. Progress estimators provide a principled means for deciding when a behavior may be terminated even if its goal is not reached and an externally generated event has not occurred. Fortuitous reward refers to reward ascribed to a particular situation–behavior (or state–action) pair which is actually a result of previous behaviors/actions. It manifests as follows: previous behaviors lead the system near the goal, but some event induced a behavior switch, and subsequent achievement of the goal is ascribed most strongly to the final behavior, rather than the previous ones. Shaped reward in the form of progress estimators effectively eliminates this effect. Because it provides feedback during behavior execution, it rewards the previous beneficial behaviors more strongly than the final one, thus more appropriately assigning credit. This case therefore illustrates how reinforcement learning has been successfully applied to behavior-based robotics, in particular at the level of behavior selection. The learning process is accelerated by the behavior structure, which provides a higher-level representation of actions and time-extended dynamics.

13.5.2 Learning Behavior Networks

The modularization of behavior-based systems as networks of behaviors allows for learning to be applied at the network level as well. *Nicolescu* and *Matarić* [13.40, 41] developed the notion of *abstract behaviors*, which separate the activation conditions of a behavior from its output actions (so-called *primitive behaviors*, which share the same principles as basis behaviors described Sect. 13.3); this allows for a more general set of activation conditions to be associated with the primitive behaviors. While this is not necessary for any single task, it is what provides generality to the representation. An abstract behavior is a pairing of a given behavior's activation conditions (i. e., preconditions) and its effects (i. e., postconditions); the result is an abstract and general operator much like those used in classical deliberative systems (Fig. 13.2). Primitive behaviors, which typically consist of a small basis set, may involve one or an entire collection of sequential or concurrently executing behaviors.

Networks of such behaviors are then used to specify strategies or general *plans* in a way that merges the advantages of both abstract representations and behavior-based systems. The nodes in the networks are abstract behaviors, and the links between them repre-

sent precondition and postcondition dependencies. The task plan or strategy is represented as a network of such behaviors. As in any behavior-based system, when the conditions of a behavior are met, the behavior is activated. Analogously, when the conditions of an abstract behavior are met, the behavior activates one or more primitive behaviors which achieve the effects specified in its postconditions. The network topology at the abstract behavior level encodes any task-specific behavior sequences, freeing up the primitive behaviors to be reused for a variety of tasks. Thus, since abstract behavior networks are computationally lightweight, solutions for multiple tasks can be encoded within a single system and dynamically switched.

Nicolescu and *Matarić* [13.40, 41] introduced a means for automatically generating such networks offline as well as at runtime. The latter enables a learning robot to acquire task descriptions dynamically, while observing its environment, which can include other robots and/or a teacher. The methodology was validated on a mobile robot following a human and acquiring a representation of the human-demonstrated task by observing the activation of its own abstract behavior pre- and postconditions, thus resulting in a new abstract behavior network representing the demonstrated task [13.76]. The robot was able to acquire novel behavior sequences and combinations (i. e., concurrently executing behaviors), resulting in successful learning of tasks involving visiting various targets in particular order as shown in VIDEO 27 and VIDEO 28, picking up, transporting, and delivering objects as shown in VIDEO 32 and VIDEO 33, dealing with barriers, and maneuvering obstacle courses in specific ways as shown in VIDEO 30 and VIDEO 31.

This approach has been extended to learning of more general task representations, in which every goal is achieved through a linear superposition (or fusion) of robot primitives, and sequencing across goals is achieved through arbitration [13.77]. In this work, controllers are built from two components: *behavior primitives* (BPs) and *fusion primitives* (FPs), which can be sequenced to form *behavior networks*. The *behavior primitives* perform a set of actions under given environmental conditions and are represented using a schema-based approach [13.78]. A *fusion primitive* encapsulates a set of multiple concurrently running primitive behaviors through linear combination of the motor commands, expressed as a vector in the robot's coordinate system. The FPs combine these vectors by weighting them and fusing them through vector addition. The method allows for learning of both the goals involved in the task (represented as the sequence of goal-achieving fusion primitives) and also of the particular ways in which the same goals can be achieved (captured in the behavior weights). Using this learning approach, a robot is capable of learning multiple navigation styles (such as staying on the center/left/right side of a corridor, preferences for turning left or right at T-junctions and particular styles of navigating around obstacles).

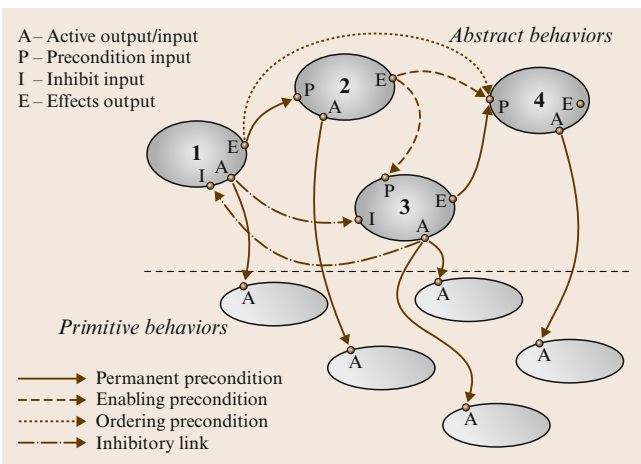


Fig. 13.2 Behavior network

13.5.3 Learning from Demonstration in Behavior-Based Systems

Learning by demonstration has also been combined with behavior-based systems to produce one-shot learning and teaching mechanisms for robot control [13.40, 79]. Furthermore, this type of behavior-based architecture has been used to learn ship navigation strategies [13.80]; during a learning phase, an instructor selects behaviors for a ship to execute in order to reach a specific goal, and a subsequent offline stage then generates dependency links between the behaviors that it witnessed during the learning phase.

Another form of learning by demonstration has used probabilistic methods to select and combine multiple behaviors [13.81]. There, the problem of learning what behaviors to execute during autonomous navigation is treated as a state estimation problem. During a learning phase, the robot observes commands used by a teacher. A particle filter then fuses the control commands that were demonstrated by the teacher to estimate behavior activation. The method produces a robust controller that is well suited for dynamic environments.

Learning by demonstration has also been used in a case-base reasoning framework [13.82], to teach a 4-legged Aibo robot different low-level navigation behaviors, such as following a ball and reaching a target. The learned behaviors encapsulate the kinematics and dynamics of the robot's control through the instruction process, which would be more difficult to achieve if de-

Avidbots Ex1016
Page 359 of 2154

signed by hand, and can successfully be combined into a newer, higher-level behavior, thus enabling hierarchical representations of robot tasks.

A method based on decision trees and Support Vector Machines has been used in [13.83] to find mappings between sensory data and patterns produced by behaviors in a learned feature space. The mapping allows a robot to identify behaviors shown by a human demonstrator and to further learn a sequential representation of the demonstrated task.

13.5.4 Learning from History of Behavior Use

Most deliberative approaches derive knowledge for reasoning about the world from sensor inputs and actions taken by the robot. This results in complex state-space representations of the world and does not take into consideration the context in which these sensations/actions are taken. As already discussed, behaviors, which are readily used as low-level control blocks driven by what is experienced from the interactions with the environment, can also serve as an abstract representation to model those interactions. One approach is to use history information [13.84], i. e., to explicitly take into consideration the time sequence of observations in order to make a decision. Applying this idea to behavior-based systems, by knowing the purpose of each of the behaviors and by observing their history of use, the robot can reason and ground its intentions in what it is experiencing in its operating environment. The concept of abstract behavior is exploited here as well, to activate behaviors and as a representation on which to learn.

Learning from history has been validated in behavior-based systems capable of making the robot change its behavior selection strategy for foraging colored objects (blocks) to a homing region in nonstationary, dynamically changing environments involving multiple concurrently learning robots [13.49–51]. In that foraging task, the robot is given two subtasks: search for a block (searching task), and bring it to the home region (homing task). The robot is given behaviors to accomplish these tasks: one behavior for the searching task, called *searching-block*, and two for the homing task, *homing* and *drop-block*. A *velocity control* behavior is also used in both of these tasks to make the robot move. All these behaviors are referred to as *task behaviors*. Conditions for activating task behaviors are preprogrammed based on the presence or absence of a block in front of the robot and the proximity of the home region.

The robot also needs to navigate safely in the environment. In this approach, an *avoidance* behavior is activated unless the robot is near home and carrying a block; otherwise it is disabled to allow the robot to approach the home region. This type of behavior, used for handling harmful situations and interference while accomplishing a task, is referred to as a *maintenance behavior*. The designer determines the conditions in which maintenance behaviors should be used, but cannot indicate when they will occur during the task, as that is tied to the dynamics of the interaction between the robot and its environment.

The robot learns to use *alternative behaviors* (*follow-side*, *rest*, and *turn-randomly*), which introduce variety into its action repertoire by changing the way it accomplishes its tasks. In contrast to other types of behaviors, these have no a priori activation conditions; the objective is to allow the robot to learn when to activate these behaviors according to past experiences, for some preset periods of time, when it is experiencing interference in accomplishing its task. Figure 13.3 illustrates how the behaviors are prioritized using a fixed suppression mechanism, similar to the subsumption architecture [13.9] but with the difference that the activated behaviors, i. e., those allowed to issue outputs, change dynamically.

Following behavior-based principles, an activated behavior may or may not be used to control the robot,

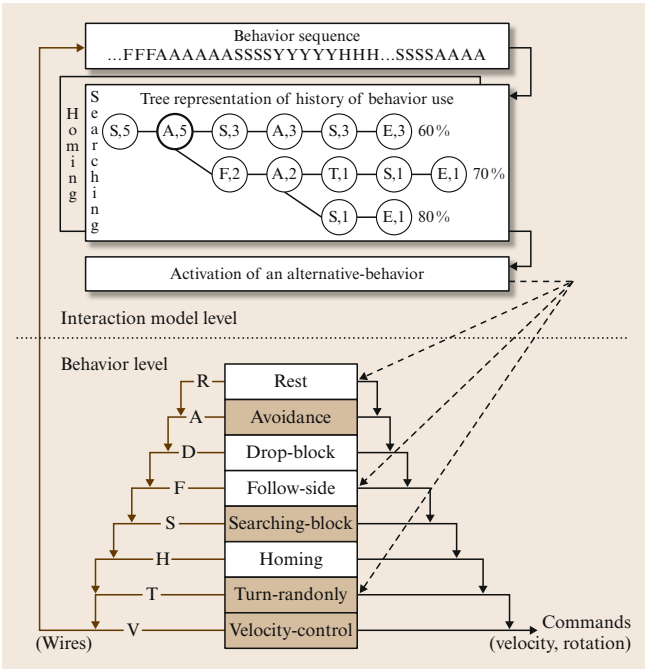


Fig. 13.3 Organization of the behavior level and the interaction model level. Behavior on *colored background* represent an example of activated behaviors for the searching task, with *Turn-randomly* as a chosen alternative behavior. For clarity, sensory inputs to behaviors are not shown

Avidbots Ex1016
Page 360 of 2154

depending on the sensory conditions it monitors and the arbitration mechanism. An activated behavior is used only when it provides commands that actually control the robot. Whenever a behavior is used, its corresponding symbol is sent to the interaction model, generating the sequence of behaviors used over time. Separate learning trees are used for each task; determining which one to use is done based on the activated task behavior.

The algorithm uses a tree structure to store the history of behavior use. The upper part of Fig. 13.3 shows a typical tree with nodes storing the behavior (H for use by *homing* and *drop-block*) used for controlling the robot while accomplishing its task, and n, the number of times a transition between the node itself and its successor has been made (as observed from behavior use). Initially, the tree for a particular task is empty and is incrementally constructed as the robot goes about its task. Leaf nodes are labeled with E (for end-node) and store the total *performance* of the particular tree path. Whenever a path is completely reused, when the same sequence of behaviors is used, the E node is updated with the average of the stored and current performances for recent trials.

Learning is done through reinforcement. Depending on the domains and the tasks, a variety of factors can be used to evaluate performance, and different metrics can be used with this learning algorithm. To see how far the idea of self-evaluation and learning by observing behavior use can be taken, the evaluation function used here is not based on characteristics about the environment or the task. Instead, it is based on the amount of time behaviors are used. Comparison between the time spent using behaviors associated with the tasks and the time spent exploiting maintenance behaviors is used to derive the evaluation criterion. Consequently, behavioral selection strategy is derived from what can be learned from the experiences of the robot in its environment, without having to characterize a priori what can be considered to be optimal operating conditions in the environment.

Using the tree and the evaluation function, the algorithm has two options for using a maintenance behavior:

1. Not to make any changes in its active behavior set (the *observe* option);
2. To activate an alternative behavior.

The sequence of nodes in a tree path characterizes the interactions experienced by the robot in the world. Different selection criteria can be used by comparing the performance at the current position in the tree with the expected performance, following an exploration (to learn the effects of alternative behaviors) then exploitation (to exploit what was learned in previous trials) strategy. The expected performance of a given node is the sum of the stored end-node performances in its subpaths, multiplied by the frequency of use of the subpaths relative to the current position in the tree. Finally, since this algorithm is used in noisy and nonstationary conditions, deleting paths is necessary to keep the interaction model up to date. This is done by keeping only a fixed number of the most recent paths in the tree.

Results obtained with this approach show that the robot is able to learn unanticipated strategies (like resting in front of a static obstacle to increase the turning angle when it starts to move again, making it locate a target) and original ones (like yielding when close to other robots or following walls when the center of the pen is crowded). Developing such capabilities is important in general, because it makes it possible for robots to learn in nonstationary environments, which are common in natural settings.

13.6 Applications and Continuing Work

Behavior-based robots have demonstrated various standard capabilities, including obstacle avoidance, navigation, terrain mapping, following, chasing/pursuit, object manipulation, task division and cooperation, and learning maps [13.85], navigation and walking. They have also demonstrated some novel applications like large-scale group behaviors including flocking, foraging, and soccer playing, human–robot interaction, and modeling insect [13.86] and even human behavior [13.87–89]. Behavior-based methods span mobile robots, shopping carts [13.25], underwater vehicles, planetary rovers for space exploration, assistive [13.90], interactive and social robots [13.91, 92], as well as snake-like modular robots [13.93], robots capable of grasping, manipulation, walking, running, and many others. Consumer market products, such as the iRobot Roomba Robotic Floorvac [13.94], also use behavior-based control, demonstrating its broad applicability.

The use of behavior-based architectures for robot control has evolved from single-system implementations to approaches that combine forms of learning, state estimation, and distributed computation. Behaviors have been combined with fuzzy inference systems for indoor navigation using mobile robots [13.35, 36, 95–97], where a command fusion module acts as an arbiter that combines multiple fuzzy behavior outputs

Avidbots Ex1016
Page 361 of 2154

into a single control signal. This strategy ensures the robot is capable of making inferences in the face of uncertainty.

Behavior-based methods have been employed in multi-robot systems from the outset [13.43], as illustrated by VIDEO 34, and are still used [13.98, 99]. Multi-robot researchers also consider tasks requiring tightly coupled cooperation; [13.100] for an overview and discussion. Such tasks typically require low-level sensor-sharing and/or for higher-level explicit coordination. Behavior-based controllers have been developed and extended in order to address these challenges. For example, *Parker* et al. [13.101] considered reusable behavior units that can be automatically redistributed for low-level information processing. *Werger* and *Matarić* [13.102] described broadcast of local eligibility (BLE) to enable higher-level group behaviors by allowing communications to influence each robot's local action selection mechanism. *Gerkey* and *Matarić* [13.103, 104] demonstrated scalable and efficient market-based coordination algorithms for multi-robot coordination in a variety of tasks, including tightly coupled ones (e.g., box-pushing [13.105]). A similar strategy was applied to a robot sensor network, which achieves concurrent, sequential and asynchronous multi-robot task allocation [13.106]. The market-based auction algorithm developed in this system uses a fitness function that enhances fault tolerance and the fairness of the task allocation process. *Sedat* and *Aydan* [13.107] use a fractal conductivity-based approach to generate movements of a coordinated team of mobile sensors. The team explores an environment to reach a target that is a source of a potential field chemical substance contaminated into sea or lake water.

Several researchers have shown that behavior-based controllers allow for sophisticated coordination through a utility-centered model of the collective task. Behaviors use this representation to produce actions that consider each robot's impact on the performance of the group as a whole. *Iocchi* et al. [13.108] have shown this in heterogeneous multi-robot system, while *Batalin* and *Sukhatme* [13.109] demonstrated that complex, interrelated and dynamic tasks can be performed in coordinated ways in robots with behavior-based controllers interacting with a sensor network. *Stroupe* and *Balch* [13.110] considered a mapping task and developed move value estimation for robot teams (MVERT), essentially a behavior-based method for maximizing group knowledge.

Because behavior-based methods lend itself naturally to multi-robot control problems, they have had a significant influence on that field of research. *Simmons* et al. [13.111] described a hybrid architecture designed for group-level coordination which employs behaviors as a method for organizing low-level safety-critical controller code. Behaviors have also been used to structure controllers and networked communication in minimalist systems [13.112]. Some multi-robot research with a control-theoretic flavor has used separate executable processes that can be switched on and off dynamically depending on task constraints, in a manner very similar to behavior-based control [13.113].

Behavior-based controllers have also been built using genetic algorithms [13.114], in which the controller learned through coevolution five behavioral components (waypoint prediction, force field trajectory, speed regulation, reversing and heading alignment), enabling it to outperform all its opponents in a real time car racing game held during the 2007 IEEE Congress on Evolutionary Computation (CEC).

Behavior-based architectures have also been used in complex vision systems for identification rather than control. In those contexts, each behavior represents a small unit of visual computation, such as frame differencing, motion detection, and edge detection, resulting in biologically inspired vision and attention behavior [13.115, 116].

Behavior-based architectures have also been developed for the control and learning in humanoid robots. In the past, the Cog project demonstrated behavior-based control for articulated manual and eye–hand coordination [13.117, 118]. *Edsinger* [13.119] developed a lightweight behavior architecture for organizing perception and control of Domo. The architecture allows for specification of time-contingent behaviors and distributed computation, resulting in a real-time controller that allows Domo to operate in an environment with humans. Kismet [13.120] involved several behavior-based systems that controlled perception, motivation, attention, behavior, and movement in a human–robot interaction (HRI) context. Each behavior represented Kismet's individual drives and motivations. Situated modules and episode rules were introduced as part of an HRI architecture by *Ishiguro* et al. [13.121] and *Kanda* et al. [13.122]; they employed an ordered set of general-use situated modules and behaviors that are contingent on a condition, used in sequences defined by a set of task-specific episode rules.

13.6.1 Adaptive Behavior Selection – A Case Study

Designing robots capable of safely and effectively interacting with humans in real-world settings is one of the ultimate challenges in robotics. Representation and abstract reasoning are necessary to keep up with and adapt to the inherently complex properties of such environments and tasks.

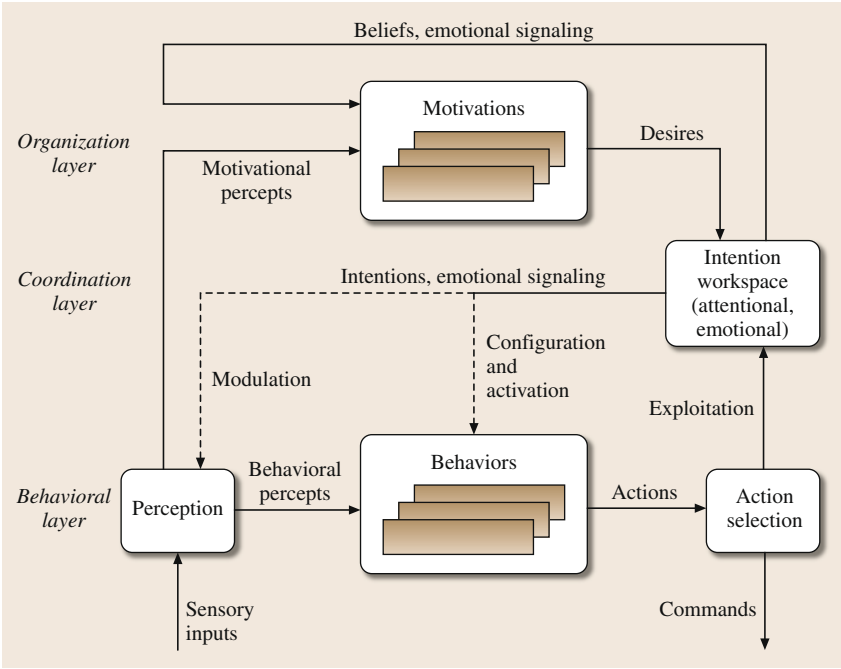


Fig. 13.4 Hybrid Behavior-Based Architecture (HBBA)

Adaptation in behavior-based systems [13.28] generally consists of modifying internal parameters of the behaviors, or modifying the set of activated behaviors. Many possible avenues to implement such mechanisms exist. One possibility consists of adding to a behavior-based architecture the idea of intentionally activating, monitoring, and configuring behaviors. An architectural framework based on this idea has evolved over the years, having been implemented on diverse robotic platforms and tested in various operational conditions:

- EMIB – Emotion, Motivation and Intentional Behaviors – has been applied to robots with different capabilities, such as a robot that uses activation variables, topological localization and mapping, and fuzzy behaviors to explore and characterize an environment [13.45, 46], and on an autonomous rolling robot that only uses simple sensors and a microcontroller to generate purposeful movements for conducting studies regarding on toddler–robot interaction [13.123, 124].
- MBA – Motivated Behavioral Architecture [13.125, 126]). In the American Association for Artificial Intelligence (AAAI) mobile robot competition and as shown in VIDEO 417, MBA was used with the Spartacus robot to demonstrate how a behavior-based system could integrate planning and sequencing tasks under temporal constraints, interleaving planning and execution [13.127] (as in [13.128] and [13.129]), and with spatial localization capabilities using a previously generated metric map. The system also used behavioral message reading [13.130], sound processing capabilities with an eight-microphone system for source localization, tracking, real-time sound separation [13.131, 132], and a touch screen interface to allow the robot to acquire information about where it is in the world, what it should do, and how it should do it [13.125].

HBBA – Hybrid Behavior-Based Architecture [13.133] – illustrated by Fig. 13.4, is the current instantiation of the architectural framework. The Behavioral Layer consists of behaviors that are activated and configured according to what are referred to as the Intentions of the system (derived by the Motivation Layer). Intentions are associated with the activation and configuration of behaviors and with the modulation of perceptual modules. The highest layer uses Motivations acting as distributed processes (just like Behaviors) and manifesting Desires for the satisfaction or inhibition of Intentions. The Intention Workspace processes these Desires to issue Intentions. The Intention Workspace can also provide Emotional Signaling (detecting situations based on models of how Behaviors are exploited over time [13.44, 134–136]), and generates Beliefs (i. e., knowledge about the interaction dynamics of the robot and the environment) which can be used to identify conflicts between Desires and their satisfaction. The objective

Avidbots Ex1016
Page 363 of 2154

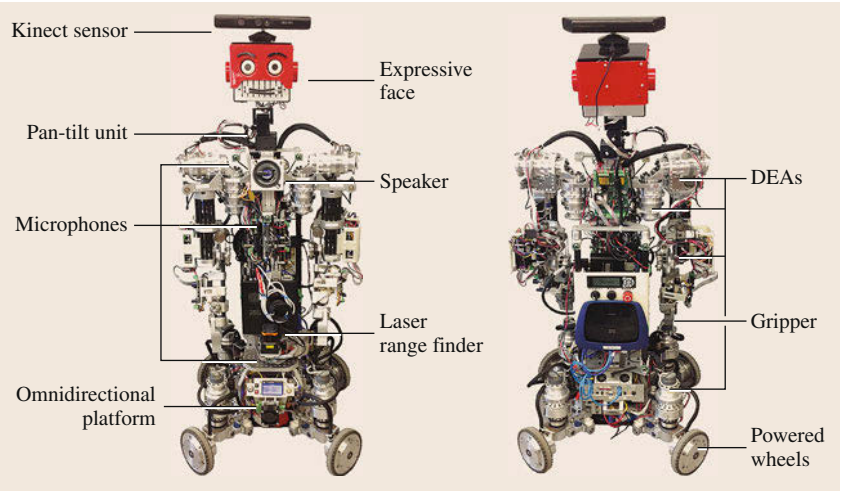


Fig. 13.5 Photographs of the front and back of the robot, IRL-1 (30 DOF and 55 sensors). DEAs stand for Differential Elastic Actuators (after [13.133])

is to have Intentions emerge from the interaction of independent motivations. Desires are communicated to the Intention Workspace, which plays a role similar to a coordination layer in a hybrid architecture, or to action selection with motivations acting as behaviors and Intentions emerging from the interaction of independent motivations. Our design requirements for the Intention Workspace module is to implement generic mechanisms (i. e., mechanisms that are independent of the robot's physical capabilities and intended usage) to generate Beliefs (i. e., data structures that represent desires, knowledge about the interaction dynamics of the robot and the environment), and identify conflicts between Desires and their satisfaction. The Exploitation link is used to derive information about the effective use of Behaviors as influenced by the events occurring in the world and the Action Selection mechanism. We believe that observing the exploitation of behaviors over time is an important source of information about the emerging functionality that comes from the Behaviors and the Motivations, because it combines both a representation of the environment (from the behavioral percepts associated with Behaviors) and of the control policy (from the internal decision-making processes).

HBBA is currently being used with IRL-1 [13.137, 138], an interactive omnidirectional platform shown in Fig. 13.5, VIDEO 418 and VIDEO 419.

IRL-1 has behaviors for safe navigation, recharge and social interaction (with gestures, voice and vision). Action selection is priority-based. Motivations are implemented to provide the intrinsic modes of operation of the robot (safe navigation while exploring the world, interact with people and ensuring energetic autonomy), to localize and plan paths to desired locations, and eventually to plan and schedule tasks [13.139]. Just like action selection at the behavior-based level, the Intention Workspace plays a key role. More specifically, in HBBA, it involves:

- A tree representation of desires. Each desire can be associated with a particular configuration and activation of one or more behaviors. The tree representation of desires provides a general representation of Desires' interdependencies issued by the independent motivations, from high-level/abstract desires (e.g., *Deliver message*) to primitive/behavior-related desires (e.g., *Avoid obstacles*), allowing for behavior configurations to arise in a distributed fashion based on the capabilities available to the robot. This allows motivations to exchange information asynchronously on how to activate, configure, and monitor behaviors. For instance, one motivation may monitor the robot's energy level to issue a recharging desire, which would be associated with a *Recharge* behavior that would make the robot opportunistically detect and dock to a charging station. Meanwhile, if the robot knows where it is and can determine a path to a nearby charging station, a path planning motivation could add a sub-desire of navigating to this position, using a *Goto* behavior. If a person is detected, a social motivation may provide an alternative desire by engaging into a conversation to ask for guidance toward the nearest charging station. The robot would then have three alternative solutions to fulfill its desire to recharge.
- A representation of the history of behavior exploitations (similar to one described in Sect. 13.5.4), indexed by intentions. For instance, for the recharging desire example, if being guided by people reveals not to be as reliable as using the internal map, a preference could be given to the second option.

Avidbots Ex1016
Page 364 of 2154

- An attention selection mechanism to module computational load, by percepts selection or output complexity reduction. For instance, artificial audition [13.131, 132] can be modulated to detect sounds, localize sound sources or separate sound sources depending on the presence of people surrounding the robot and its intentions to interact. A Simultaneous Localization and Mapping (SLAM) algorithm can change resolution depending on available processing load of onboard computers.
- An emotion-based mechanism to regulate conflicts between desires and intentions. Robot's adaptability in dynamic, continuous and unpredictable environments ultimately depends on the detection of the situations for which their regular decision-making process is not appropriate. Psychologists have identified that one of the functions of human emotion is to highlight these kinds of situations, allowing other cognitive processes to address them. Therefore, an emotional process that detect such situations from the use of temporal models of intentions could be beneficial [13.44]. This process does not rely on a priori knowledge of specific environmental conditions nor of the robot's mission objectives, but only on temporal analysis of how behaviors satisfy the robot's intentions.

13.7 Conclusions and Further Reading

This chapter has described behavior-based control, a methodology for single- and multi-robot control aimed at situated robots operating in unconstrained, challenging, and dynamic conditions in the real world. While inspired by the philosophy of reactive control, behavior-based systems are fundamentally more expressive and powerful, enabling representation, planning, and learning capabilities. Distributed behaviors are used as the underlying building blocks for these capabilities, allowing behavior-based systems to take advantage of dynamic interactions with the environment rather than rely solely on explicit reasoning and planning. As the complexity of robots continues to increase, behavior-based principles and their applications in robot architectures and deployed systems will evolve as well, demonstrating increasingly higher levels of situated intelligence and autonomy.

Interested readers can find more information regarding behavior-based systems in other chapters of this Handbook, as well as in *Brooks* [13.140], *Arkin* [13.22], in artificial intelligence and robotics textbooks [13.141, 142], and in introductory textbooks on mobile robot control [13.143–145].

Video-References

VIDEO 27 Experience-based learning of high-level task representations: Demonstration
available from http://handbookofrobotics.org/view-chapter/13/videodetails/27

VIDEO 28 Experience-based learning of high-level task representations: Reproduction
available from http://handbookofrobotics.org/view-chapter/13/videodetails/28

VIDEO 30 Experience-based learning of high-level task representations: Demonstration (2)
available from http://handbookofrobotics.org/view-chapter/13/videodetails/30

VIDEO 31 Experience-based learning of high-level task representations: Reproduction (2)
available from http://handbookofrobotics.org/view-chapter/13/videodetails/31

VIDEO 32 Experience-based learning of high-level task representations: Demonstration (3)
available from http://handbookofrobotics.org/view-chapter/13/videodetails/32

VIDEO 33 Experience-based learning of high-level task representations: Reproduction (3)
available from http://handbookofrobotics.org/view-chapter/13/videodetails/33

VIDEO 34 The Nerd Herd
available from http://handbookofrobotics.org/view-chapter/13/videodetails/34

VIDEO 35 Toto
available from http://handbookofrobotics.org/view-chapter/13/videodetails/35

VIDEO 417 SpartacUS
available from http://handbookofrobotics.org/view-chapter/13/videodetails/417

VIDEO 418 Natural interaction design of a humanoid robot
available from http://handbookofrobotics.org/view-chapter/13/videodetails/418

VIDEO 419 Using ROS4iOS
available from http://handbookofrobotics.org/view-chapter/13/videodetails/419

Avidbots Ex1016
Page 365 of 2154

References

13.1 J.S. Albus: Outline for a theory of intelligence, IEEE Trans. Syst. Man Cybern. **21**(3), 473–509 (1991)

13.2 G. Girald, R. Chatila, M. Vaisset: An integrated navigation and motion control system for autonomous multisensory mobile robots, Proc. 1st Int. Symp. Robotics Res. (1983)

13.3 H. Moravec, A. Elfes: High resolution maps from wide angle sonar, Proc. IEEE Int. Conf. Robotics Autom. (1995)

13.4 J. Laird, P. Rosenbloom: An investigation into reactive planning in complex domains, Proc. 9th Natl. Conf. Am. Assoc. Artif. Intell. (1990) pp. 1022–1029

13.5 N.J. Nilsson: *Shakey the Robot*, Tech. Rep. No. 325 (SRI International, Menlo Park 1984)

13.6 S.J. Rosenschein, L.P. Kaelbling: A situated view of representation and control, Artif. Intell. **73**, 149–173 (1995)

13.7 R.A. Brooks: Elephants don't play chess. In: *Designing Autonomous Agents: Theory and Practice form Biology to Engineering and Back*, ed. by P. Maes (MIT Press, Cambridge 1990) pp. 3–15

13.8 R. Brooks, J. Connell: Asynchrounous distributed control system for a mobile robot, Proc. SPIE Intell. Control Adapt. Syst. (1986) pp. 77–84

13.9 R.A. Brooks: A robust layered control system for a mobile robot, IEEE J. Robotics Autom. **RA-2**(1), 14–23 (1986)

13.10 P.E. Agre, D. Chapman: Pengi: An implementation of a theory of activity, Proc. 6th Natl. Conf. Am. Assoc. Artif. Intell. (1987) pp. 268–272

13.11 R.A. Brooks: Intelligence without representation, Artif. Intell. **47**, 139–159 (1991)

13.12 M. Schoppers: Universal plans for reactive robots in unpredictable domains, Proc. Int. Jt. Conf. Artif. Intell. (1987) pp. 1039–1046

13.13 P.E. Agre, D. Chapman: What are plans for? In: *Designing Autonomous Agents: Theory and Practice form Biology to Engineering and Back*, ed. by P. Maes (MIT Press, Cambridge 1990) pp. 17–34

13.14 G.N. Saridis: Intelligent robotic control, IEEE Trans. Autom. Control **AC-28**(5), 547–557 (1983)

13.15 R.J. Firby: An investigation into reactive planning in complex domains, Proc. AAAI Conf. (1987) pp. 202–206

13.16 R. Arkin: Towards the unification of navigational planning and reactive control, Proc. Am. Assoc. Artif. Intell., Spring Symp. Robotics Navig. (1989) pp. 1–5

13.17 C. Malcolm, T. Smithers: Symbol grounding via a hybrid architecture in an autonomous assembly system. In: *Designing Autonomous Agents: Theory and Practice from Biology to Engineering and Back*, ed. by P. Maes (MIT Press, Cambridge 1990) pp. 123–144

13.18 J.H. Connell: SSS: A hybrid architecture applied to robot navigation, Proc. IEEE Int. Conf. Robotics Autom. (1992) pp. 2719–2724

13.19 E. Gat: Integrating planning and reacting in a heterogeneous asynchronous architecture for controlling real-world mobile robots, Proc. Natl. Conf. Artif. Intell. (1992) pp. 809–815

13.20 M. Georgeoff, A. Lansky: Reactive reasoning and planning, Proc. 6th Natl.Conf. Am. Assoc. Artif. Intell. (1987) pp. 677–682

13.21 B. Pell, D. Bernard, S. Chien, E. Gat, N. Muscettola, P. Nayak, M. Wagner, B. Williams: An autonomous spacecraft agent prototype, Auton. Robots **1-2**(5), 1–27 (1998)

13.22 R.C. Arkin: *Behavior-Based Robotics* (MIT Press, Cambridge 1998)

13.23 M.J. Matarić: Reinforcement learning in the multi-robot domain, Auton. Robots **4**(1), 73–83 (1997)

13.24 P. Bonasso, R.J. Firby, E. Gat, D. Kortenkamp, D.P. Miller, M.G. Slack: Experiences with an architecture for intelligent reactive agents, Proc. Int. Jt. Conf. Artif. Intell. (1995)

13.25 M. Goller, T. Kerscher, J.M. Zollner, R. Dillmann, M. Devy, T. Germa, F. Lerasle: Setup and control architecture for an interactive shopping cart in human all day environments, Proc. Int. Conf. Adv. Robotics (2009) pp. 1–6

13.26 P. Pirjanian: Multiple objective behavior-based control, Robotics Auton. Syst. **31**(1-2), 53–60 (2000)

13.27 P. Pirjanian: *Behavior Coordination Mechanisms – State-of-the-Art*, Tech. Rep. IRIS-99-375 (Univ. of Southern California, Institute of Robotics and Intelligent Systems, Los Angeles 1999) pp. 99–375

13.28 M. Scheutz, V. Andronache: Architectural mechanisms for dynamic changes of behavior selection strategies in behavior-based systems, IEEE Trans. Syst. Man Cybern. B **34**(6), 2377–2395 (2004)

13.29 M. Proetzsch, T. Luksch, K. Berns: Development of complex robotic systems using the behavior-based control architecture iB2C, Robotics Auton. Syst. **58**(1), 46–67 (2010)

13.30 D. Payton, D. Keirsey, D. Kimble, J. Krozel, J. Rosenblatt: Do whatever works: A robust approach to fault-tolerant autonomous control, Appl. Intell. **2**(3), 225–250 (1992)

13.31 P. Maes: Situated agents can have goals. In: *Designing Autonomous Agents: Theory and Practice form Biology to Engineering and Back*, ed. by P. Maes (MIT Press, Cambridge 1990) pp. 49–70

13.32 P. Maes: The dynamics of action selection, Proc. Int. Jt. Conf. Artif. Intell. (1989) pp. 991–997

13.33 B.A. Towle, M. Nicolescu: Real-world implementation of an Auction Behavior-Based Robotic Architecture (ABBRA), Proc. IEEE Int. Conf. Technol. Pract. Robot Appl. (2012) pp. 79–85

13.34 B.A. Towle, M. Nicolescu: Fusing multiple sensors through behaviors with the distributed architecture, Proc. IEEE Int. Conf. Multisens. Fusion Integr. Intell. Syst. (2010) pp. 115–120

13.35 A. Saffiotti: The uses of fuzzy logic in autonomous robot navigation, Soft Comput. **1**, 180–197 (1997)

Avidbots Ex1016
Page 366 of 2154

13.36 F. Michaud: Selecting behaviors using fuzzy logic, Proc. IEEE Int. Conf. Fuzzy Syst. (1997)

13.37 E. Gat: On three-layer architectures. In: *Artificial Intelligence and Mobile Robotics*, ed. by D. Kortenkamp, R. Bonasso, R. Murphy (MIT/AAAI Press, Cambridge 1998)

13.38 M.J. Matarić: Integration of representation into goal-driven behavior-based robots, IEEE Trans. Robotics Autom. **8**(3), 304–312 (1992)

13.39 M.J. Matarić: Navigating with a rat brain: A neurobiologically-inspired model for robot spatial representation, From animals to animats. Proc. 1st Int. Conf. Simul. Adapt. Behav. (1990) pp. 169–175

13.40 M. Nicolescu, M.J. Matarić: Experience-based representation construction: Learning from human and robot teachers, Proc. IEEE/RSJ Int. Conf. Intell. Robot Syst. (2001) pp. 740–745

13.41 M. Nicolescu, M.J. Matarić: A hierarchical architecture for behavior-based robots, Proc. Int. Jt. Conf. Auton. Agents Multiagent Syst. (2002)

13.42 L.E. Parker: ALLIANCE: An architecture for fault tolerant multirobot cooperation, IEEE Trans. Robotics Autom. **14**(2), 220–240 (1998)

13.43 M.J. Matarić: Designing and understanding adaptive group behavior, Adapt. Behav. **4**(1), 50–81 (1995)

13.44 C. Raïevsky, F. Michaud: Improving situated agents adaptability using interruption theory of emotions, From animals to animats. Proc. Int. Conf. Simul. Adapt. Behav. (2008) pp. 301–310

13.45 F. Michaud, G. Lachiver, C.T. Le Dinh: Architectural methodology based on intentional configuration of behaviors, Comput. Intell. **17**(1), 132–156 (2001)

13.46 F. Michaud: EMIB–Computational architecture based on emotion and motivation for intentional selection and configuration of behaviour-producing modules, Cogn. Sci. Q. **3-4**, 340–361 (2002)

13.47 F. Michaud, M.T. Vu: Managing robot autonomy and interactivity using motives and visual communication, Proc. Int. Conf. Auton. Agents (1999) pp. 160–167

13.48 O. Petterson, L. Karlsson, A. Saffiotti: Model-free execution monitoring in behavior-based robotics, IEEE Trans. Syst. Man Cybern. **37**(4), 890–901 (2007)

13.49 F. Michaud, M.J. Matarić: Learning from history for behavior-based mobile robots in non-stationary environments, Auton. Robots **5**(3/4), 335–354 (1998)

13.50 F. Michaud, M.J. Matarić: Learning from history for behavior-based mobile robots in non-stationary environments II, Auton. Robots **31**(3-4), 335–354 (1998)

13.51 F. Michaud, M.J. Matarić: Representation of behavioral history for learning in nonstationary conditions, Robotics Auton. Syst. **29**(2), 1–14 (1999)

13.52 O.C. Jenkins, M.J. Matarić: Deriving action and behavior primitives from human motion data, Proc. IEEE/RSJ Int. Conf. Intell. Robot Syst. (2002) pp. 2551–2556

13.53 O.C. Jenkins, M.J. Matarić: Automated derivation of behavior vocabularies for autonomous humanoid motion, Proc. 2nd Int. Jt. Conf. Auton. Agents Multiagent Syst. (2003)

13.54 M.J. Matarić: Designing emergent behaviors: From local interactions to collective intelligence, From animals to animats 2. Proc. 2nd Int. Conf. Simul. Adapt. Behav. (1992) pp. 432–441

13.55 S. Saripalli, D.J. Naffin, G.S. Sukhatme: Autonomous flying vehicle research at the University of Southern California, Proc. 1st Int. Work. Multi-Robot Syst. (2002) pp. 73–82

13.56 M.J. Matarić: Behavior-based control: Examples from navigation, learning, and group behavior, J. Exp. Theor. Artif. Intell. **9**(2–3), 323–336 (1997)

13.57 P. Maes, R.A. Brooks: Learning to coordinate behaviors, Proc. 8th Natl. Conf. Artif. Intell. AAAI (1990) pp. 796–802

13.58 H. Yanco, L.A. Stein: An adaptive communication protocol for cooperating mobile robots, From animals to animats 3. Proc 3rd Int. Conf. Simul. Adapt. Behav. (1993) pp. 478–485

13.59 J.R. del Millàn: Learning efficient reactive behavioral sequences from basic reflexes in a goal-directed autonomous robot, From animals to animats 3. Proc. 3rd Int. Conf. Simul. Adapt. Behav. (1994) pp. 266–274

13.60 L. Parker: Learning in cooperative robot teams, Proc. Int. Jt. Conf. Artif. Intell. (1993) pp. 12–23

13.61 M.J. Matarić: Learning to behave socially, From animals to animats 3. Proc. 3rd Int. Conf. Simul. Adapt. Behav. (1994) pp. 453–462

13.62 M. Asada, E. Uchibe, S. Noda, S. Tawaratsumida, K. Hosoda: Coordination of multiple behaviors acquired by a vision-based reinforcement learning, Proc. IEEE/RSJ Int. Conf. Intell. Robot Syst. (1994)

13.63 J. McCarthy: Making robots conscious of their mental states, AAAI Spring Symp. (1995)

13.64 T. Smithers: On why better robots make it harder, From animals to animats. Proc. 3rd Int. Conf. Simul. Adapt. Behav. (1994) pp. 64–72

13.65 D. McFarland, T. Bösser: *Intelligent Behavior in Animals and Robots* (MIT Press, Cambridge 1993)

13.66 P. Maes: A bottom-up mechanism for behavior selection in an artificial creature, From animals to animats. Proc. 1st Int. Conf. Simul. Adapt. Behav. (1991) pp. 238–246

13.67 B.M. Blumberg, P.M. Todd, P. Maes: No bad dogs: Ethological lessons for learning in Hamsterdam, From animals to animats. Proc. Int. Conf. Simul. Adapt. Behav., ed. by P. Maes, M.J. Matarić, J.-A. Meyer, J. Pollack, S.W. Wilson (1996) pp. 295–304

13.68 C. Breazeal, B. Scassellati: Infant-like social interactions between a robot and a human caregiver, Adapt. Behav. **8**(1), 49–74 (2000)

13.69 F. Michaud, P. Pirjanian, J. Audet, D. Létourneau: Artificial emotion and social robotics. In: *Distributed Autonomous Robotic Systems*, ed. by L.E. Parker, G. Bekey, J. Barhen (Springer, Tokyo 2000) pp. 121–130

Avidbots Ex1016
Page 367 of 2154

13.70 A. Stoytchev, R. Arkin: Incorporating motivation in a hybrid robot architecture, J. Adv. Comput. Intell. Intell. Inf. **8**(3), 269–274 (2004)

13.71 S. Mahadevan, J. Connell: Automatic programming of behavior-based robots using reinforcement learning, Artif. Intell. **55**, 311–365 (1992)

13.72 J. Kober, A. Wilhelm, E. Oztop, J. Peters: Reinforcement learning to adjust parametrized motor primitives to new situations, Auton. Robots **33**, 361–379 (2012)

13.73 M.J. Matarić: Reward functions for accelerated learning, Proc. 11th Int. Conf. Mach. Learn., New Brunswick, ed. by W.W. Cohen, H. Hirsh (Morgan Kauffman, Boston 1994) pp. 181–189

13.74 H. Gleitman: *Psychology* (Norton, New York 1981)

13.75 M. Dorigo, M. Colombetti: *Robot Shaping: An Experiment in Behavior Engineering* (MIT Press, Cambridge 1997)

13.76 M. Nicolescu, M.J. Matarić: Learning and interacting in human-robot domains, IEEE Trans. Syst. Man Cybern. **31**(5), 419–430 (2001)

13.77 M. Nicolescu, O.C. Jenkins, A. Olenderski, E. Fritzinger: Learning behavior fusion from demonstration, Interact. Stud. **9**(2), 319–352 (2008)

13.78 R.C. Arkin: Motor schema based navigation for a mobile robot: An approach to programming by behavior, Proc. IEEE Int. Conf. Robotics Autom. (1987) pp. 264–271

13.79 S.B. Reed, T.R.C. Reed, M. Nicolescu, S.M. Dascalu: Recursive, hyperspherical behavioral learning for robotic control, Proc. IEEE World Autom. Congr. (2010) pp. 1–8

13.80 A. Olenderski, M. Nicolescu, S. Louis: A behavior-based architecture for realistic autonomous ship control, Proc. IEEE Symp. Comput. Intell. Games (2006)

13.81 M. Nicolescu, O.C. Jenkins, A. Olenderski: Learning behavior fusion estimation from demonstration, Proc. IEEE Int. Symp. Robot Hum. Interact. Commun. (2006) pp. 340–345

13.82 J.M. Peula, C. Urdiales, I. Herrero, I. Sánchez-Tato, F. Sandoval: Pure reactive behavior learning using case based reasoning for a vision based 4-legged robot, Robotics Auton. Syst. **57**(67), 688–699 (2009)

13.83 S. Huang, E. Aertbelien, H. Van Brussel, H. Bruyninckx: A behavior-based approach for task learning on mobile manipulators, Proc. 41st Int. Symp. Robotics 6th Ger. Conf. Robotics (2010) pp. 1–6

13.84 A.K. McCallum: Hidden state and reinforcement learning with instance-based state identification, IEEE Trans. Syst. Man Cybern. B **26**(3), 464–473 (1996)

13.85 E. Jauregi, I. Irigoien, B. Sierra, E. Lazkano, C. Arenas: Loop-closing: A typicality approach, Robotics Auton. Syst. **59**(3-4), 218–227 (2011)

13.86 D.J. Harvey, T.-F. Lu, M.A. Keller: Comparing insect-inspired chemical plume tracking algorithms using a mobile robot, IEEE Trans. Robotics **24**(2), 307–317 (2008)

13.87 A. Agha, G. Bekey: Phylogenetic and ontogenetic learning in a colony of interacting robots, Auton. Robots **4**(1), 85–100 (1997)

13.88 R.A. Brooks, L. Stein: Building brains for bodies, Auton. Robots **1**(1), 7–25 (1994)

13.89 B. Webb: Robotic experiments in cricket phonotaxis, From animals to animats 3. Proc. 3rd Int. Conf. Simul. Adapt. Behav. (1994) pp. 45–54

13.90 C. Liu, K. Conn, N. Sakar, W. Stone: Online affect detection and robot behavior adaptation for intervention of children with autism, IEEE Trans. Robotics **24**(4), 883–896 (2008)

13.91 N. Mitsunaga, C. Smith, T. Kanda, H. Ishiguro, N. Hagita: Adapting robot behavior for human–robot interaction, IEEE Trans. Robotics **24**(4), 911–916 (2008)

13.92 J.-W. Yoon, S.-B. Cho: An intelligent synthetic character for smartphone with Bayesian networks and behavior selection networks, Expert Syst. Appl. **39**(12), 11284–11292 (2012)

13.93 A. Brunete, M. Hernando, E. Gambao, J.E. Torres: A behaviour-based control architecture for heterogeneous modular, multi-configurable, chained micro-robots, Robotics Auton. Syst. **60**(12), 1607–1624 (2012)

13.94 J.L. Jones: Robots at the tipping point, IEEE Robotics Autom. Mag. **13**(1), 76–78 (2006)

13.95 M.F. Selekwa, D.D. Dunlap, D. Shi, E.G. Collins Jr.: Robot navigation in very cluttered environments by preference-based fuzzy behaviors, Robotics Auton. Syst. **56**(3), 231–246 (2008)

13.96 P. Rusu, E.M. Petriu, T.E. Whalen, A. Coronell, H.J.W. Spoelder: Behavior-based neuro-fuzzy controller for mobile robot navigation, IEEE Trans. Instrum. Meas. **52**(4), 1335–1340 (2003)

13.97 R. Huq, G.K.I. Mann, R.G. Gosine: Behaviour modulation technique in mobile robotics using fuzzy discrete event system, IEEE Trans. Robotics **22**, 903–916 (2006)

13.98 J.S. Cepeda, L. Chaimowicz, R. Soto, J.L. Gordillo, E.A. Alanís-Reyes, L.C. Carrillo-Arce: A behavior-based strategy for single and multi-robot autonomous exploration, Sensors **12**(9), 12772–12797 (2012)

13.99 A. Marino, L. Parker, G. Antonelli, F. Caccavale: A decentralized architecture for multi-robot systems based on the null-space-behavioral control with application to multi-robot border patrolling, J. Intell. Robotic Syst. **71**(3-4), 423–444 (2013)

13.100 L.E. Parker: Current research in multirobot systems, Artif. Life Robotics **7**(1–2), 1–5 (2003)

13.101 L.E. Parker, M. Chandra, F. Tang: Enabling autonomous sensor-sharing for tightly-coupled cooperative tasks, Proc. 1st Int. Work. Multi-Robot Syst. (2005) pp. 119–230

13.102 B.B. Werger, M.J. Matarić: Broadcast of Local Eligibility for Multi-Target Observation, Proc. 5th Int. Conf. Distrib. Auton. Robotics Syst. (2000) pp. 347–356

13.103 B.P. Gerkey, M.J. Matarić: Principled communication for dynamic multi-robot task allocation, Lect. Notes Control Inform. Sci. **271**, 353–362 (2001)

Avidbots Ex1016
Page 368 of 2154

13.104 B.P. Gerkey, M.J. Matarić: Sold!: Auction methods for multi-robot coordination, IEEE Trans. Robotics Autom. **18**(5), 758–768 (2002)

13.105 B.P. Gerkey, M.J. Matarić: Pusher-watcher: An approach to fault-tolerant tightly-coupled robot coordination, Proc. IEEE Int. Conf. Robotics Autom. (2002) pp. 464–469

13.106 B. Akin, A.M. Aydan, I. Erkmen: A behavior based layered, hybrid, control architecture for robot/sensor networks, Proc. IEEE Int. Conf. Robotics Autom. (2006) pp. 206–211

13.107 N. Sedat, E. Aydan: A fractal conductivity-based approach to mobile sensor networks in a potential field, Int. J. Adv. Manuf. Technol. **37**(7), 732–746 (2008)

13.108 L. Iocchi, D. Nardi, M. Piaggio, A. Sgorbissa: Distributed coordination in heterogeneous multi-robot systems, Auton. Robots **15**(2), 155–168 (2004)

13.109 M. Batalin, G. Sukhatme: Coverage, exploration and deployment by a mobile robot and communication network, Telecommun. Syst. **26**(2–4), 181–196 (2004)

13.110 A.W. Stroupe, T. Balch: Value-based action selection for observation with robot teams using probabilistic techniques, Robotics Auton. Syst. **50**(2–3), 85–97 (2005)

13.111 R. Simmons, T. Smith, M.B. Dias, D. Goldberg, D. Hershberger, A. Stentz, R. Zlot: A Layered Architecture for coordination of mobile robots, Proc. NRL Work. Multi-Robot Syst. (2002)

13.112 J. Nembrini, A. Winfield, C. Melhuish: Minimalist coherent swarming of wireless networked autonomous mobile robots, Proc. 7th Int. Conf. Simul. Adapt. Behav. (2002) pp. 373–382

13.113 M. Egerstedt, X. Hu: Formation constrained multi-agent control, IEEE Trans. Robotics Autom. **17**(6), 947–951 (2001)

13.114 C.H. Tan, K.C. Tan, A. Tay: Computationally efficient behaviour based controller for real time car racing simulation, Expert Syst. Appl. **37**(7), 4850–4859 (2010)

13.115 K. Gold, B. Scassellati: Learning about the self and others through contingency, AAAI Spring Symp. Dev. Robotics (2005)

13.116 M. Baker, H.A. Yanco: Automated street crossing for assistive robots, Proc. Int. Conf. Rehabil. Robotics (2005) pp. 187–192

13.117 M. Williamson: Postural primitives: Interactive behavior for a humanoid robot arm, Proc. Int. Conf. Simul. Adapt. Behav. (1996)

13.118 M. Marjanovic, B. Scassellati, M. Williamson, R. Brooks: The Cog Project: Building a humanoid robot, Lect. Notes Artif. Intell. **1562**, 52–87 (1998)

13.119 A. Edsinger: Robot Manipulation in Human Environments, Ph.D. Thesis (Massachusetts Institute of Technology, Cambridge 2007)

13.120 C. Breazeal: Infant-like social interactions between a robot and a human caretaker, Adapt. Behav. **8**(1), 49–74 (2000)

13.121 H. Ishiguro, T. Kanda, K. Kimoto, T. Ishida: A robot architecture based on situated modules, Proc. IEEE/RSJ Int. Conf. Intell. Robot Syst. (1999) pp. 1617–1623

13.122 T. Kanda, T. Hirano, D. Eaton, H. Ishiguro: Person identification and interaction of social robots by using wireless tags, Proc. IEEE/RSJ Int. Conf. Intell. Robotics Syst. (2003) pp. 1657–1664

13.123 F. Michaud, J.F. Laplante, H. Larouche, A. Duquette, S. Caron, D. Létourneau, P. Masson: Autonomous spherical mobile robotic to study child development, IEEE Trans. Syst. Man. Cybern. **35**(4), 1–10 (2005)

13.124 F. Michaud, S. Caron: Roball, the rolling robot, Auton. Robots **12**(2), 211–222 (2002)

13.125 F. Michaud, C. Côté, D. Létourneau, Y. Brosseau, J.-M. Valin, É. Beaudry, C. Raïevsky, A. Ponchon, P. Moisan, P. Lepage, Y. Morin, F. Gagnon, P. Giguère, M.-A. Roux, S. Caron, P. Frenette, F. Kabanza: Spartacus attending the 2005 AAAI Conference, Auton. Robots **12**(2), 211–222 (2007)

13.126 F. Michaud, Y. Brosseau, C. Côté, D. Létourneau, P. Moisan, A. Ponchon, C. Raïevsky, J.-M. Valin, E. Beaudry, F. Kabanza: Modularity and integration in the design of a socially interactive robot, Proc. IEEE Int. Work. Robot Hum. Interact. Commun. (2005) pp. 172–177

13.127 E. Beaudry, Y. Brosseau, C. Côté, C. Raïevsky, D. Létourneau, F. Kabanza, F. Michaud: Reactive planning in a motivated behavioral architecture, Proc. Am. Assoc. Artif. Intell. Conf. (2005) pp. 1242–1247

13.128 K. Haigh, M. Veloso: Planning, execution and learning in a robotic agent, Proc. 4th Int. Conf. Artif. Intell. Plan. Syst. (1998) pp. 120–127

13.129 S. Lemai, F. Ingrand: Interleaving temporeal planning and execution in robotics domains, Proc. Natl. Conf. Artif. Intell. (2004) pp. 617–622

13.130 D. Létourneau, F. Michaud, J.-M. Valin: Autonomous robot that can read, EURASIP J. Appl, Signal Process. **17**, 1–14 (2004)

13.131 J.-M. Valin, F. Michaud, J. Rouat: Robust localization and tracking of simultaneous moving sound sources using beamforming and particle filtering, Robotics Auton. Syst. **55**(3), 216–228 (2007)

13.132 J.-M. Valin, S. Yamaoto, J. Rouat, F. Michaud, K. Nakadai, H.G. Okuno: Robust recognition of simultaneous speech by a mobile robot, IEEE Trans. Robotics **23**(4), 742–752 (2007)

13.133 F. Michaud, F. Ferland, D. Létourneau, M.-A. Legault, M. Lauria: Toward autonomous, compliant, omnidirectional humanoid robots for natural interaction in real-life settings, Paladyn. J. Behav. Robotics **1**, 57–65 (2010)

13.134 D.O. Hebb: *The Organization of Behavior: A Neuropsychological Theory* (Wiley, New York 1949)

13.135 G. Mandler: *Mind and Body: Psychology of Emotion and Stress* (Norton, New York 1984)

13.136 J.E. Stets: Emotions and sentiments. In: *Handbook of Social Psychology*, ed. by J.E. DeLamater (Wiley, New York 2003)

13.137 F. Ferland, F. Ferland, D. Létourneau, M.-A. Legault, M. Lauria, F. Michaud: Natural interaction design of a humanoid robot, J. Hum.-Robot Interact. **1**(2), 118–134 (2012)

Avidbots Ex1016
Page 369 of 2154

13.138 F. Ferland, R. Chauvin, D. Létourneau, F. Michaud: Hello Robot, can you come here? Using ROS4iOS to provide remote perceptual capabilities for visual location, speech and speaker recognition, Proc. ACM/IEEE Conf. Hum.-Robot Interact. (1983) p. 101
13.139 E. Beaudry, D. Létourneau, F. Kabanza, F. Michaud: Reactive planning as a motivational source in a behavior-based architecture, Proc. IEEE/RSJ Int. Conf. Intell. Robot Syst. (2008) pp. 1848–1853
13.140 R.A. Brooks: *Cambrian Intelligence – The Early History of the New AI* (MIT Press, Cambridge 1999)
13.141 R. Pfeifer, C. Scheier: *Understanding Intelligence* (MIT Press, Cambridge 2001)
13.142 R.R. Murphy: *An Introduction to AI Robotics* (MIT Press, Cambridge 2000)
13.143 M.J. Matarić: *The Robotics Primer* (MIT Press, Cambridge 2007)
13.144 F. Martin: *Robotic Explorations: A Hands-On Introduction to Engineering* (Prentice Hall, Upper Saddle River 2001)
13.145 J.L. Jones, A.M. Flynn: *Mobile Robots – Inspiration to Implementation* (Peters, Wellesley 1993)

Avidbots Ex1016
Page 370 of 2154

Multimedia Contents

14. AI Reasoning Methods for Robotics

Michael Beetz, Raja Chatila, Joachim Hertzberg, Federico Pecora

Artificial intelligence (AI) reasoning technology involving, e.g., inference, planning, and learning, has a track record with a healthy number of successful applications. So can it be used as a toolbox of methods for autonomous mobile robots? Not necessarily, as reasoning on a mobile robot about its dynamic, partially known environment may differ substantially from that in knowledge-based pure software systems, where most of the named successes have been registered. Moreover, recent knowledge about the robot's environment cannot be given a priori, but needs to be updated from sensor data, involving challenging problems of symbol grounding and knowledge base change.

This chapter sketches the main robotics-relevant topics of symbol-based AI reasoning. Basic methods of knowledge representation and inference are described in general, covering both logic- and probability-based approaches. The chapter first gives a motivation by example, to what extent symbolic reasoning has the potential of helping robots perform in the first place. Then (Sect. 14.2), we sketch the landscape of representation languages available for the endeavor. After that (Sect. 14.3), we present approaches and results for several types of practical, robotics-related reasoning tasks, with an emphasis on temporal and spatial reasoning. Plan-based robot control is described in some more detail in Sect. 14.4. Section 14.5 concludes.

14.1 **Why Should a Robot Use AI-Type Reasoning?** 330
14.2 **Knowledge Representation and Processing** 330
14.2.1 Logic-Based Reasoning 331
14.2.2 Probabilistic Reasoning 334
14.3 **Reasoning and Decision Making** 338
14.3.1 Using Multiple Knowledge Representation Formalisms 339
14.3.2 Reasoning About Time 339
14.3.3 Reasoning About Space 342
14.3.4 Reasoning About Multiple KR Formalisms 344
14.3.5 Knowledge Representation Systems for Robots 345
14.4 **Plan-Based Robot Control** 346
14.4.1 Models of Plan-Based Control 346
14.4.2 Representation of Plans 348
14.4.3 Generating Robot Action Plans 349
14.4.4 Revising and Improving Plans 350
14.4.5 Other Reasoning Methods 350
14.5 **Conclusions and Further Reading** 351
14.5.1 Further Reading 351
Video-References 351
References 352

Historical debates about the necessity and wisdom of employing symbolic reasoning in autonomous mobile robots notwithstanding (Chap. 13), there seems to be agreement now that some part or layer in the control system of such robots should or could include reasoning. Hybrid control architectures (Chap. 12) would be typical software structures for solving the difficult problem of amalgamating their contributions with those of other parts of the controller, yet guaranteeing sufficiently short control cycle times that the robot can act safely in a dynamic environment.

Symbolic reasoning is understood here in the classical sense of artificial intelligence (AI), i. e., deliberation based on symbols as in first-order predicate logic (FOPL) or Bayesian probability theory, but typically with restrictions and/or extensions thereof, which have to be traded off against each other to achieve a suitable combination of expressivity and inference speed.

Avidbots Ex1016
Page 371 of 2154

14.1 Why Should a Robot Use AI-Type Reasoning?

If robots are to accomplish tasks that are too complicated for specifying the appropriate behavior for all possible situations beforehand, they should be able to reason themselves about what is the appropriate way to perform the task – even if the task is specified vaguely. In this setting, *reasoning* means inferring the needed knowledge from what is spelled out explicitly in the available knowledge. It further means *automated* reasoning using methods from AI including knowledge representation (KR) and inference, heuristic search, and machine learning.

To perform their course of action appropriately, robots need to reason about the actions they intend to perform, their intended effects, the unwanted side effects, whether they are executable in the respective situations, the goals the actions are to accomplish, and so on. Consider, for example, the seemingly simple task of picking up an object from a table. To do so, the robot has to decide where to go in order to pick up the object, which hand(s) to use, how to reach for the object, which type of grasp to apply, where to place the gripper, how much grasp force to apply, how much lift force, how to lift the object, where to hold it, and so on.

If programmers have to specify the decisions for every conceivable combination of object and task, the control program for picking up an object becomes very complex. But even this program would in most cases not suffice to generate competent robot behavior, because how the pickup action should be performed depends on the context, too – the state of the object, the task to be performed, the scene that the object is located in. If the object is a glass filled with juice, it has to be held upright and if it is a bottle the robot intends to fill a glass with, it should not grasp the top of the bottle. If the bottle is to be picked up in the middle of a cluttered scene, then grasping the top of the bottle might be the best choice. If the scene is cluttered and the purpose is filling a glass, then the robot might even have to re-grasp after picking up the object.

The situation gets even more complicated if the task is not a single action, but a compound activity such as cleaning the table. In this case, AI reasoning methods can be employed to enable programmers to specify the course of activity very compactly: *for every object on the table put it where it belongs*. For such a vague plan to be a sufficient activity prescription, the robot has to infer the information needed to decide on the appropriate parameterizations of the pickup actions and it has to infer where objects belong depending on whether they are to be used again, are dirty, perishable, etc. Competently cleaning the table might also require the robot to think about the order in which it wants to put away the objects. It might decide to group the carry actions according to the place where the items have to be placed, or clean up perishable items first. It might reason about whether to stack items, get a tray and use that, and whether to leave doors of cupboards open.

Thus, an important task of AI reasoning techniques is: given a vague instruction, infer what is the appropriate action, what is the appropriate object to be acted on, what is the appropriate order of action executions, and what is the appropriate way of performing every action.

To deal with these issues, autonomous robots can represent and reason about aspects including the robots' capabilities, their environments, the objects they are to act on, their actions and the effects they cause, and other agents in the environment.

Important kinds of reasoning that robots should be capable of, therefore include the following:

- *Prediction* (often called *temporal projection*): inferring what will (probably) happen if the intended course of action is executed
- *Envisioning*: inferring (all) possible events and effects that will happen if a plan gets executed in a hypothetical situation
- *Diagnosis*: inferring what caused a particular event or effect in plan execution
- *Query answering*: given some knowledge preconditions for plan execution (e.g., a robot has to know the combination of a safe in order to open it), inferring pieces of knowledge that satisfy these knowledge preconditions.

14.2 Knowledge Representation and Processing

Reasoning requires that the reasoner – in our case, a robot – has an explicit representation of parts or aspects of its environment to reason about. Two questions come up immediately: what are suitable formats for such an explicit representation, and where does the represented knowledge come from?

The second question refers to the problem of generating and maintaining in real time a symbolic description of the robot's environment, or at least of some part of it, based on, possibly, a previous symbolic description and on recent environment information as by sensors and by communication with other agents. In full

Avidbots Ex1016
Page 372 of 2154

generality, this problem is currently unsolved, involving AI fundamentals such as *symbol grounding* [14.1] and *object anchoring* [14.2]. So practical symbolic reasoning in a robot is restricted to that part of its knowledge that can be kept sufficiently recent. This includes, obviously, static knowledge about the environment, such as the topological elements and their relations in a building; transient knowledge available in symbolic forms, such as in facility management data bases; and, most challenging, symbolic data that has to be distilled from sensor data. Object recognition (Chap. 33) is a crucial ingredient to solve this; these issues will not be treated here.

This section deals with answers to the first question, i. e., formalisms suitable for representing knowledge. Suitability has two aspects here that are to be treated as two sides of one coin. One is *epistemological adequacy*: Does the formalism allow the targeted aspects of the environment to be expressed compactly and precisely? The other is *computational adequacy*: Does the formalism allow typical inferences to be drawn effectively and efficiently? There is a tradeoff between the two adequacies, given that very rich, expressive, and hence epistemologically appealing formalisms are typically accompanied by intractable or even undecidable inference problems, and vice versa. KR, then [14.3, p. xiii]:

> *is the field of AI that focuses on the design of formalisms that are both epistemologically and computationally adequate for expressing knowledge about a particular domain.*

The plural in *formalisms* does not come by chance: there is no such thing as *the* KR language. The reason is twofold.

First, the archetype of KR languages, FOPL, is undecidable, i. e., any reasoner dealing with a language at least as expressive as FOPL is not guaranteed to terminate on a single query, let alone terminate fast. However, full FOPL expressivity is not always needed; for example, to represent a finite domain or to make restricted use of quantification may lead to decidable, and even tractable representation languages. Second, some applications of KR and reasoning, like quite a number of applications in Robotics, require evidence rather than truth to be represented, and FOPL, in particular, does not come handy for doing that. So, there are good reasons, both for pragmatics and for epistemological adequacy, for having several representation languages – and that has happened in AI.

Table 14.1 lists in its rows four broad types of KR languages, which have been in use in Robotics. Description logics (DLs) are in fact a specific family of relational languages, of which there are decidable and tractable members. The table gives in its columns the *Assertions* of the language types, i. e., the language elements used to represent some domain; the *Queries*, i. e., the type of problems that a reasoner in a particular language is supposed to solve; the *Subject* that a particular language of the respective type is normally used for representing; and *Examples* of such languages. The remainder of this section sketches generic KR in logic and in a probabilistic language, including the respective types of inferences. Specific robotics-related instances of reasoning on the symbol level are described in the following section, in particular, reasoning about temporal and spatial objects and relations, which are pervasive in a robot's reasoning.

14.2.1 Logic-Based Reasoning

As stated above, FOPL does not qualify for a KR formalism in the above-mentioned sense of being both epistemologically and computationally adequate for typical application domains – in many cases, it is neither of the two. Yet, it is the background for understanding conceptually and mathematically general relational languages.

In this short text, we can only introduce KR in logic by way of example, assuming familiarity with the basic notions of FOPL. For more comprehensive and formal treatments, we refer to the broad body of literature: every typical AI textbook introduces FOPL, and [14.4, 5] are no exceptions; [14.6] is a principled introduction to logics; [14.7] a practical one, and [14.8] a concise mathematical one.

Table 14.1 Abstract categorization of main relational languages as used in Robotics (see text for explanations)

	Assertions	Queries	Subject	Example
General relational languages	Sentences, theories	(Logical) consequence	Situations, contexts, relations	FOPL, prolog, situation calculus
Description logics	A-Box and T-Box contents	Subsumption, consistency	Ontologies, encyclopedias	OWL-DL
Probabilistic languages	Prior and conditional probabilities	Conditional probabilities	Evidence	Bayesian networks
Languages for temporal and spatial reasoning	Temporally and/or spatially qualified sentences	Logical consequence, satisfiability, minimal representation	Desired or observed spatial/temporal state of affairs	LTL, IA, TCSP, RCC, ARA$^+$

Avidbots Ex1016
Page 373 of 2154

Representing knowledge in some variant of logic facilitates to draw inferences from that knowledge using provably sound and/or complete calculi. Automated deduction is a subfield of logic and AI that offers a healthy number of very powerful implemented deduction systems that can readily be used. Reference [14.4, Chap. 9] gives an introduction; [14.9] is a traditional handbook covering all the classical topics of logical reasoning.

Relying on logical inference, a robot could deduce a large number of facts that might otherwise be hard or impossible to get at. For example, assume a robot perceives by acquiring and interpreting sensor data, that the door D_{509} in an office building is currently closed: $Closed(D_{509})$ expressed in some ad hoc FOPL language, assuming intuitive interpretations of symbols. Let us further assume that the robot's base of static knowledge about the building contains the sentences

$$\begin{aligned}
&Connects(D_{509}, C_5, R_{509})\ ,\\
&Connects(D_{508}, C_5, R_{508})\ ,\\
&Connects(D_{508a}, R_{508}, R_{509})\ ,\\
&\forall d, l_1, l_2.[Connects(d, l_1, l_2) \leftrightarrow\\
&\quad Connects(d, l_2, l_1)]\ ,\\
&\forall d.[Closed(d) \leftrightarrow \neg Open(d)]\ ,\\
&\forall l.[At(l) \rightarrow Acessible(l)]\ ,\\
&\forall l_1, l_2.[Accessible(l_1) \rightarrow\\
&\quad (\exists d.[Connects(d, l_1, l_2) \wedge Open(d)] \rightarrow\\
&\quad Accessible(l_2))]\ . \qquad (14.1)
\end{aligned}$$

Constants D_i and variable d denote doors; R_i denote rooms; C_i corridors; variables l, l_i denote locations, i. e., rooms and corridors. Assume that the robot's localization tells it its current room or corridor location as $At(\cdot)$.

Then, assuming the robot knows it is $At(C_5)$, observing $Closed(D_{509})$ entails $\neg Open(D_{509})$, and, more interestingly, that $Accessible(R_{509})$ is true only if $Open(D_{508}) \wedge Open(D_{508a})$ is true. So, performing, for example, a delivery task to room R_{509}, the robot may replan its route through R_{508}, unless at least one of D_{508}, D_{508a} is known to be closed. If the status of one or both of them is unknown (neither $Open(\cdot)$ nor $Closed(\cdot)$ is entailed by the current knowledge base), then accessibility of R_{509} can be neither proven nor disproven. That would leave open the option of planning the route through D_{508} and D_{508a}, gathering their required statuses on site.

So, as this little example may show, FOPL is a powerful representation and reasoning tool. Moreover, its theory is very well understood. In particular, it is well-known that consequence in FOPL is in general undecidable, that means, a sound and complete deduction algorithm cannot even guarantee termination for some particular inference attempt, let alone speedy results.

However, that does not mean logic as a representation and inference tool needs to be completely disposed of. Many applications do not require the full expressivity of FOPL. Moreover, there are many interesting language subsets of FOPL that are decidable and can be handled by algorithms that are efficient in most practical cases. So considerable effort by the KR community has gone into identifying FOPL subsets and fitting inference procedures that qualify for being both epistemologically and computationally adequate for a broad number of applications. We will consider two of them in more detail: propositional logic and description logics; we will then go briefly into using logics for high-level robot control.

Propositional Theories

In quite a number of practical cases, FOPL theories (sets of formulas) represent in fact finite domains. Logically speaking, they have finite Herbrand universes or can at least be recast in a form such that they have. In this case, it may still be handy to express the domain theory in FOPL syntax, but all that goes beyond a purely propositional theory is just for notational convenience. For example, an axiom stating that a robot can only be in one location (room or corridor) at a time

$$\forall l_1, l_2.[At(l_1) \rightarrow (\neg At(l_2) \vee l_1 = l_2)] \qquad (14.2)$$

can be *flattened* for a finite building into the, clumsy but equivalent, form handling all locations explicitly, for example in a six-storey building

$$\begin{aligned}
&At(R_{001}) \rightarrow [\neg At(R_{002}) \wedge \cdots \wedge \neg At(R_{514}) \wedge\\
&\quad \neg At(C_0) \wedge \cdots \wedge \neg At(C_5)]\ ,\\
&At(R_{002}) \rightarrow [\neg At(R_{001}) \wedge \cdots \wedge \neg At(R_{514}) \wedge\\
&\quad \neg At(C_0) \wedge \cdots \wedge \neg At(C_5)]\ ,\\
&\ldots \qquad (14.3)
\end{aligned}$$

where every ground (variable-free) instance of a predicate, such as the At instances above, are to be considered as propositional variables, regarding textual identity.

The good news here is that the corresponding *flat* theory to a FOPL theory over a finite Herbrand universe is propositional, hence, it is decidable. Moreover, under some practical conditions – e.g., if the variables in the FOPL theory are sorted (that is, the information is available that the variables l_1, l_2, e.g., range over rooms and corridors) – it can even be generated mechanically from a more compact FOPL syntax.

The potentially bad news is that the now propositional theory may of course consist of a huge amount

Avidbots Ex1016
Page 374 of 2154

of propositional sentences: in general, as all combinations of variable substitutions in all FOPL sentences need to be generated, the growth is multiexponential in terms of the domain sizes of the FOPL variables.

However, the technology for propositional satisfiability checking or *model checking* is making considerable progress, allowing propositional theories with thousands or even tens of thousands of variables to be handled in the order of seconds of computation time on regular hardware. The respective methods are particularly efficient if many models exist for a satisfiable propositional formula or if the truth or falsity of many ground facts is known a priori (such as $At(C_5)$ for a robot by independent localization). Both is often the case in practical KR.

References [14.4, Chap. 7.6] and [14.9, Chap. 24] give introductions to model checking. One modern family of methods is based on the classical Davis–Putnam (DPLL) algorithm [14.10]. It attempts to construct systematically a propositional model of a given theory, efficiently propagating interpretation constraints for variables. Another family of algorithms applies local search techniques, attempting to generate interpretations using random (Monte Carlo) variable assignments. Reference [14.11] is a Web page maintained for the annual satisfiability checking competitions held together with the annual SAT (*Theory and Applications of Satisfiability Testing*) conferences; it gives a continuously updated overview over current top-performing satisfiability checking systems.

Description Logics

DL have emerged as rigorous formalizations of somewhat intuitive KR forms of the AI of the 1970s, namely, semantic networks and frame systems. Logically speaking, DLs, of which there is a considerable variety, form a certain family of subsets of FOPL, some of which are decidable or even tractable.

There are two parts of representing knowledge about a particular domain using a DL language. First, the *upper ontology* of the domain is formulated. It introduces the general domain concepts as well as relations between these concepts. A particularly interesting type of relations occurring in all ontologies is the superclass-subclass relation. Given that most DLs – as part of the means for enforcing decidability – strictly disallow cyclic relations between concepts to be specified, the superclass relation imposes a hierarchical taxonomy on concepts, which serves for defining property inheritance, much like in object-oriented programming. This first part of a DL-based domain representation is, for historical reasons, often called the *T-Box* (or *terminological knowledge*).

A *concept* in the DL language corresponds to a unary predicate. To give an example, an ontology of robot navigation domains might include concepts like *Door*, *Location*, and so on. The concept hierarchy is built by defining concept equality $=$ or a subconcept property, e.g., $Room \sqsubseteq Location$, and $Corridor \sqsubseteq Location$. Concepts can be combined using concept conjunction, disjunction, and negation ($\sqcap, \sqcup, \neg$, respectively), allowing, e.g., concept definitions like $Location = Room \sqcup Corridor$, $Door = Closed \sqcup Open$, and $Open = \neg Closed$.

Roles in a DL language correspond to binary predicates, such as *leadsTo* for a door and a location. Inversity, intersection, and union of roles are defined as expected, where $leadsTo = leadsFrom^{-1}$ is an example for defining inverse roles. Roles can be composed, such as in defining $adjacent = leadsFrom \circ leadsTo$ (location l is adjacent to m if and only if some door connects them). Finally, concepts and roles can be combined for defining new concepts and roles. In particular, it is possible to quantify over *role-fillers*, i. e., the individual objects (see next) that can be consistently substituted for role arguments. For example, one could define $BlockedLoc = Location \sqcap \neg\exists leadsFrom.Open$ (assuming the intuitive binding rules of the operators). Different variants of DLs differ in what operators they make available; the set of available operators and additional constraints on definitions shape both the expressivity and the computability of the respective DL variant – the spectrum ranges from undecidable to tractable. See [14.3] for details.

As the second part of domain representation using DLs, individual objects have to be introduced into the language of concepts and roles. This part of the domain representation is called *A-Box* (or *assertional knowledge*). For example, $Room(R_{509})$, $leadsTo(D_{509}, R_{509})$, and $Closed(D_{509})$ could be asserted.

DLs have a number of reasoning services to offer, which are based on logical inference in a given T-Box and A-Box. They include consistency of the concept definition, subsumption and disjointness of concepts, consistency of the A-Box wrt. the T-Box, concept and role instances, all of which are decidable in many DLs. Given the T-Box and A-Box rudiments above, it would, e.g., be concluded that everything is consistent and that $BlockedLoc(R_{509})$ (note that only door D_{509} is known here to lead to R_{509}!). These DL inferences are theoretically intractable, but run efficiently in most of practical cases.

Reference [14.3] provides a comprehensive overview of DL. In 2004, the WWW consortium (W3C) has defined the Web Ontology Language OWL [14.12] as a technical basis for the Semantic Web, which was followed by OWL 2 in 2009 [14.13].

Avidbots Ex1016
Page 375 of 2154

Part of the language is OWL-DL, a classical DL in the sense just described. OWL-DL ontologies are publicly available over the Web. Reference [14.14] gives a tutorial introduction.

Logics for High-Level Robot Control

Robot domains are dynamic by nature, including at least one physical agent, namely, the robot. Capturing that in logic-based formalisms is possible and has been undertaken, as introduced in [14.4, Chap. 12.3]. However, it poses some conceptual and technical problems, which we will describe at the end of this section on Logics.

Very briefly, one of them comes from the requirement of expressing concisely and reasoning efficiently with logical models of events in general, and actions in particular. Logically speaking, an individual action changes the truth value of a limited number of facts, leaving everything else as it was. For example, modeling on some abstract level the action of going from one location to another as atomic, it changes the robot's current location before and after; depending on the modeling, it may also change the battery state and the mileage count; but it does not change the layout of the building or the name of the president. The problem of formalizing actions concisely in a logical language so that they allow facts to be inferred efficiently that may or may not have changed after applying a sequence of actions, has been termed the *frame problem*. It has received much attention in the literature; there are now a number of practical solutions to it.

Another problem concerns knowledge-base update made necessary by independently changing facts. Consider, for example, a robot sitting – as it is told by its self-localization – in front of some door D, which is believed to be open, but the robot perceives a closed door. Logically, this is a contradiction. Now there are in theory several ways to make the knowledge and the perception consistent. One is to assume that D has been closed since learning that it was open – probably the most intuitive explanation. Logically just as good would be, e.g., that the perception is faulty, or that the robot has been teleported in front of a known closed door. Among these *explanations*, some are more intuitively plausible than others; logically, some would require less formulas of the knowledge base to be withdrawn and should therefore be preferred. Ideally, after replacing an old piece of information with a new one, one would have to make sure that the consequences of a retracted formula are no longer believed.

Theoretically, these problems are arbitrarily difficult. Practically, they can be sufficiently restricted to allow solutions within a neat logical framework. Typical solutions would introduce some notion of state or holding period for formulas, following the classical example of the situation calculus [14.15]. That allows change to be tracked. Typical solutions would also give up completeness of inference, resorting to a Prolog-like inference engine. Three examples for such solutions with reported applications in robot control are GOLOG [14.16], event calculus [14.17] and FLUX [14.18]. Another family of solutions would model the course of time in a more fine-grained way than switching between states. Temporal logics are appropriate here, and the temporal action logic (TAL) is an example, which has even been integrated in a performant planner [14.19]. It leads to a more sophisticated form of temporal reasoning that will be treated in Sect. 14.3.

14.2.2 Probabilistic Reasoning

KR formalisms based on logics are worth considering whenever factual knowledge is to be represented, from which consequences are to be queried. However, part of the knowledge that a robot might use about its environment does not really have this character.

Uncertainty is one of these traits, or rather, a whole family of them, as uncertainty is in itself an overloaded concept. *Lack of knowledge* is one of its aspects. Logics can handle this insofar as truth or falsity of some facts may remain undetermined. If too much is undetermined in a knowledge base, logics will no longer be able to make interesting deductions though, as *everything is possible* logically. Yet, the different possibilities may differ considerably in likelihood, according to intuition. The point here is to represent and reason about *evidence*.

The field of KR has adopted Bayesian probability as a means for representing and reasoning with evidence, using correlations between facts rather than strict implications. Note that this approach amalgamates different sources of lack of precision and completeness of knowledge. Some piece of knowledge may be unknown because it is in principle unknowable, or because it was judged too costly to build a precise theory or determine all information relevant for making a sound deduction. In either case, working with probabilities rather than binary truth values can serve as an approximation. Note that the notion of truth is still the same here as in classical logics: *objectively*, a fact is supposed to be either true or false; probability just models the *subjective* evidence that the fact is true.

We next describe two popular and powerful representation and processing formats for probabilistic knowledge: Bayesian networks (BNs) and Markov decision processes (MDPs). In analogy to Sect. 14.2.1, we here assume familiarity with the basic notions of probability theory. Reference [14.4, Chap. 13] gives an

excellent basic introduction; for a more comprehensive treatment, there is a large variety of introductory textbooks around, [14.20] being an example.

Bayesian Networks

Inference in Bayesian probability theory basically means to infer the probability of some event of interest, given prior and dependent other relevant probabilities. Practically, an important type of probabilistic inference is *diagnostic reasoning* from observed effects back to hidden causes, given rules that specify conditional probabilities from causes to effects. So the problem is, for a potential cause C and an observed effect E: Given prior probabilities $P(C)$ and $P(E)$ and the conditional probability $P(E|C)$, determine the posterior $P(C|E)$. The solution is of course given by the Bayes rule as

$$P(C|E) = \frac{P(E|C)P(C)}{P(E)} \,. \tag{14.4}$$

However, like in logical reasoning, the theoretically appealing principle turns out to be impractical if applied naively. Consider that not just one effect E may be observed, but $E_1, \ldots, E_n$; moreover, not all of them are conditionally independent. A generalized form of the Bayes rule to calculate the correct posterior is straightforward, but who can specify all the conditional probabilities involved – in the worst case $O(2^n)$ of them, where n may easily range in the hundreds in practical cases?

Until the late 1980s, this problem was more or less circumvented. One way to do so was to treat the E_i in bad faith as independent: simply take the n individual conditional probabilities $P(E_i|C)$ and use them for approximating straightforward the full joint probability distributions. Reference [14.4, Chap. 14] reviews this and other alternative approaches.

The solution used ever since it has appeared [14.21] is *Bayesian networks* (BN), more recently subsumed under the more general concept of *graphical models*. The idea is to represent the random variables as nodes in a directed acyclic graph, where a node is directly preceded by a set of parent nodes if and only if it is directly conditionally dependent on the corresponding parent variables. So the huge full joint probability distribution is broken down into many, typically very small, *local joint probability distributions* without loss of information, the trick being to use the typically many known conditional independences among variables to reduce the representational and inferential effort drastically.

Figure 14.1 shows a simple BN expressing that D is dependent on B and C, with probabilities given by a conditional probability table which specifies locally the joint probability distribution. In addition, the structure says that D is independent from A, given B and C (a node is independent from its ancestors, given its parents), and that B is independent from C, given A, i. e., $P(C|A,B) = P(C|A)$ and $P(B|A,C) = P(B|A)$.

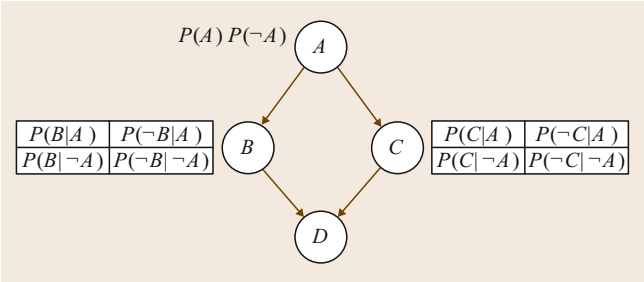


Fig. 14.1 Structure of a simple Bayesian network. It is associated with conditional probability tables. (The one for D, dependent on B and on C, is omitted)

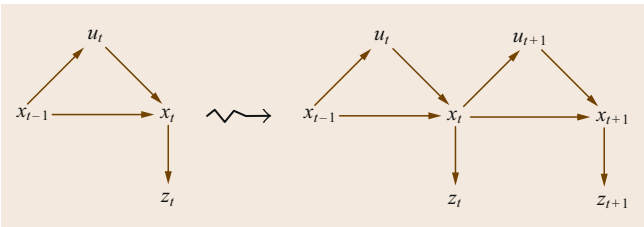


Fig. 14.2 Unfolding of a simple DBN over two time slices representing two successive states. Variables x denote states, u denote actions, and z denote measurements. The structure of each individual slice (*signaled by identical variable subscripts*) is invariant over time

Network inference can be interpreted in both directions. Bottom up, a BN enables to explain an observed phenomenon using the known conditional probabilities (diagnosis). For example, if D is observed, the likelihoods of its known causes (B or C) can be inferred. Top down, a BN enables to propagate evidence to compute the probability of a phenomenon, e.g., to compute the probability of D given A (causation).

For systems evolving over time, for which the Markov property holds, i. e., a state of a system variable in some state depends on no variables earlier than the previous state, a BN representation takes on a particular form, a *dynamic Bayesian network*, DBN. Assuming that the dependences among variables within a state remain constant over time, and assuming further that a variable x at different states is represented as a family of variables $x_1, \ldots, x_t, \ldots$, a DBN unfolded into a BN has a particular structure, as sketched by example in Fig. 14.2. The variables representing a new state correspond to a new BN *slice*, which reproduces the structure of the previous state representation and whose nodes depend only on each other and on nodes represent-

Avidbots Ex1016
Page 377 of 2154

ing the previous state. This locality allows inference in DBNs to be efficient.

Markov Decision Processes

In the context of robot control, uncertainty is induced by perception, action, and incompleteness of prior knowledge about the environment, all of which are related. Perception uncertainty results from sensor noise, from occlusion, from interpretation ambiguities, and other effects. Action uncertainty results from approximate action models (e.g., wheel slippage), unsuccessful action (e.g., grasping failure), or other practically unavoidable effects.

Figure 14.3 depicts this uncertainty laden interaction between the robot and its environment.

Probabilistic representations are used to cope with these uncertainties. Planning being a sequential decision problem, MDP and, in the case of partial observability of the environment, partially observable Markov decision processes (POMDPs) [14.22] are widely used to address them. The Markov assumption, which is the hallmark of this approach, is that a state s_n of the system depends only on the previous state s_{n-1} and the action that lead to s_n, and not on any earlier states.

Under this assumption, an MDP is formally defined by four components $\langle S, A, T, R\rangle$ where S is a finite set of states, A is a finite set of actions, $T : S \times A \to S$ the transition function defining the probability of state change upon application of a given action at time t

$$T(s, a, s') = P(s_{t+1} = s' \mid s_t = s, a_t = a) \,. \qquad (14.5)$$

T is known and provided as a table of transition probabilities. Finally, $R(s, a)$, $R : S \times A \to \mathbb{R}$, is defined as the *reward* received by the system after achieving action a leading to state s. Often, the reward is associated with the state only ($R(s)$). In the sequential process, the rewards are supposed to be additive. Solving an MDP is an optimization problem in which the sum of the expected rewards, called the *utility*, is maximized. MDPs suppose that the rewards are already known. When this is not the case, the robot can acquire them through experience. One technique to do so is reinforcement learning, as described in Chap. 15. Figure 14.4 shows an example of a simple MDP.

There are two main methods to solve an MDP. In the *value iteration* (VI) method, the first step is to compute the utilities $U(s)$ for all states, and the second step is to compute the optimal *policy* $\pi^*(s)$ which provides, for each state, the optimal action. Computing the utilities iteratively is based on the Bellman equation [14.23]

$$U_{i+1}(s) = R(s) + \gamma \max_a \sum_{s'} T(s, a, s') U_i(s') \,, \qquad (14.6)$$

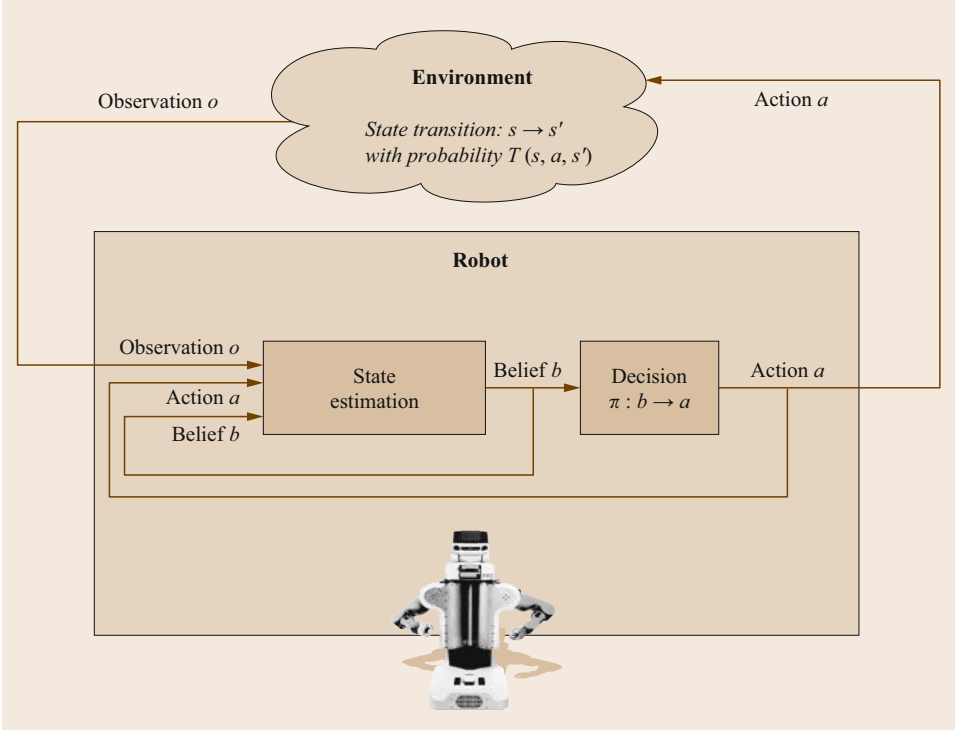


Fig. 14.3 The robot acts on its environment producing a new state with a given probability. Knowledge of the robot about state results from its observation and is also uncertain. Its decisions are taken based on its beliefs

Avidbots Ex1016
Page 378 of 2154

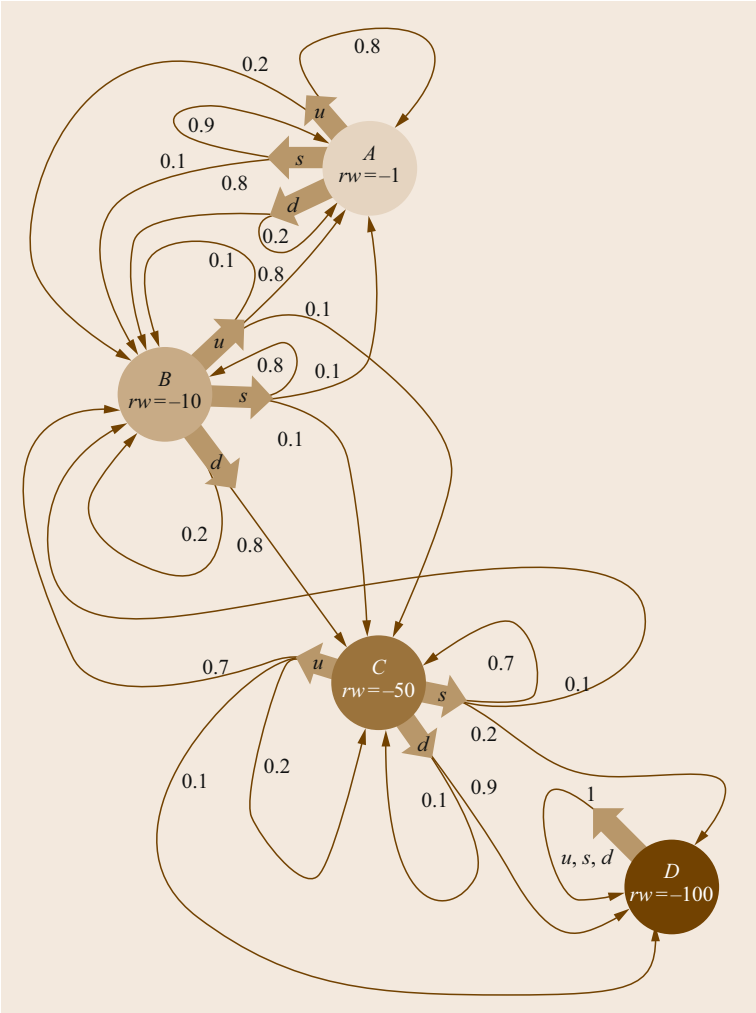


Fig. 14.4 A representation of an MDP. In this example, states A, B, C, D have associated rewards (-1, -10, -50, -100). There are three possible actions in each state, u, s, d, each have a given probability (*shown on the arrows*) to lead to another state

where $0 < \gamma < 1$ is a discount factor. After computing the utilities, the optimal policy is the one that provides the best action (i. e., the one maximizing the utility) for each state. This is computed by $\pi^*(s) = \arg\max_a \sum_{s'} T(s,a,s')U(s')$. So the result of the MDP is the best local decision. A policy is an a priori computation of the best sequence of actions, which is guided by the maximization of a global reward instead of just reaching a goal with minimal cost, as in classical planning. So a policy is not a plan of actions that has to be strictly executed. Since it already takes uncertainties in action execution into account, the policy can be pursued whatever the outcome of action execution.

The second method is called *policy iteration* (PI). Here one starts with an initial (e.g., random) policy π and, through an iterative algorithm, tries to improve it gradually by looking for the actions that augment utility.

In conclusion, in a deterministic setting, the solution to a planning problem is a sequence of actions. In an observable stochastic setting, the solution is a policy defined by the best local decision.

Partially Observable Markov Decision Processes

In the (realistic) case where the robot has no full access to the environment, decision making must be made under partial observability. To that end, the MDP formalism is enriched with an observation function that provides uncertain knowledge. A POMDP is defined by six components $\langle S, A, T, R, \Omega, O\rangle$. The first four describe an MDP. Ω is a finite set of observations o. $O : S \times A \to \Omega$ is the observation model $O(s', a, o)$ (derived for the robot sensor models) that provides the probability to obtain observation o after action a which has lead to state s'. The robot *belief state* is the probabil-

Avidbots Ex1016
Page 379 of 2154

ity distribution over all states, i. e., all the probabilities of being in all the states. The belief to be in a given state is usually noted by $b(s)$. The belief is updated after an action and observation by

$$b'(s') = \alpha O(s', a, o) \sum_s T(s, a, s') b(s) \,, \qquad (14.7)$$

where α is a normalization constant.

Although partial observability is a typical case for robots acting in mundane environments, and hence POMDPs are the more adequate formalism for modeling action and observation, POMDP-based models are often avoided. The reason is that calculating optimal POMDP policies is impracticably complex. Mostly, solving a POMDP is based on approximate or derived methods, such as working with the corresponding *belief MDP* of a POMDP, which results from considering the (fully observable) belief function induced by the POMDP.

A good example of using Markov processes for modelling and solving robotics decision-making problems, combining task planning, and motion planning is provided in [14.24].

14.3 Reasoning and Decision Making

The main motivation for developing reasoning capacities in robotics is to enable robots to make decisions about their future actions. This is why planning was one of the main early research topics in AI, within the SHAKEY project (VIDEO 704), resulting in the seminal STRIPS planner [14.25]. Designing a planning system requires to address three main questions:

- How to represent the world?
- How to represent actions?
- How to guide the plan search process?

According to the formalism used to answer these questions, the planning system will cope differently with the constraints of the real world, and this is an essential issue in robotics. As mentioned above, early planning techniques were based on first-order predicate logic (FOPL) for representing knowledge, and the limitation of FOPL when facing the uncertainties of the real world has lead to adopt probabilistic representations and probabilistic reasoning to cope with real situations.

World Representations. Classically, when reasoning at the symbolic level, knowledge about the world is represented as *states* which are sets of logical propositions such as Ontable(Cup). Under the closed world assumption, the facts that have the value FALSE are not represented in the states of the world. In order to deal with uncertainties, instead of having TRUE/FALSE values, the propositions have a probability distribution of being true.

Representation of Actions. In classical planning, as introduced by STRIPS, actions modify the state of the world and are defined by three sets of predicate lists. The predicates that should be true for the action to be feasible are the preconditions. The set of predicates that become TRUE after the action is executed is the ADD list. The set of predicates that becomes FALSE after action execution is the DELETE list. The planner includes a unification procedure to determine which predicates correspond to the propositions describing the world states. Taking into account uncertainties in action execution leads to including a probabilistic characterization of their outcome.

Search. Finding the most adequate action is based on a search algorithm in the state space. The state space is an implicit graph given by the initial state and a successor function that provides the states immediately adjacent to a given state. Each action, which represents the transition from one state to its successor, has a given cost. The well known A* algorithm [14.26] was proposed in 1968, again within the SHAKEY project, to address the search problem. The planning algorithm, based on this search, recursively selects the action that would produce the current goal state (or a subgoal thereof), and under certain conditions the optimal (i. e., least cost) solution is found.

This classical planning scheme is not able to cope with uncertainties because it is based on FOPL representations. The only way it can do that is through plan mending or replanning (Sect. 14.4). When probabilistic representations are used, the plan search will adopt a very different scheme as explained in Sect. 14.2.2.

However, reasoning in AI and robotics involves much more than sequential planning. Robots act in the real world in which action duration determines success or failure. They have to consider events occurring at given instants and situations that unfold over time. Temporal relations and an appropriate temporal logic are necessary to represent and reason about time. Section 14.3.2 discusses temporal reasoning.

Another central issue is reasoning about space. What does it mean exactly when we represent a piece of knowledge by the predicate Ontable(Cup), or Near(Robot,Table)? How to express spatial relationships symbolically from sensor data? This involves

Avidbots Ex1016
Page 380 of 2154

specific formalisms as well which are developed in Sect. 14.3.3.

14.3.1 Using Multiple Knowledge Representation Formalisms

Automated planning carries an immediate appeal for robotic applications: the actions a robot should perform to achieve a goal are obtained as a result of reasoning in a model which can be changed to suit different environments, physical capabilities, and tasks. Yet reasoning about action per se is not a silver bullet for robot programming. As noted earlier, obtaining competent robot behavior by reasoning in a model crucially depends on the epistemological adequacy of the modeling language. As a case in point, let us elaborate on our initial example: imagine a waiter robot with two arms that has to serve meals to guests in a restaurant. The robot would have to reason about time in order to infer that cold beverages should be delivered before they get warm. Food needs to be served in front of guests, hence the robot should be capable of some form of spatial reasoning. It should be capable of reasoning about resources, e.g., infer that it cannot hold more than two parts of the meal simultaneously with its two arms, or that its tray can only accommodate a limited number of dishes. A DL ontology might inform the robot that all mugs in the cupboard are clean – hence, committing to a particular mug at planning time is not necessary – or that a cup or a mug both work well for serving coffee. Most of this knowledge is difficult or impossible to model in *classical* planning models [14.27], which are limited to describing the causal relations that exist between actions and predicates in FOPL.

The types of knowledge a robot should possess ultimately depend on the application. It should be however evident that – short of encoding this knowledge in FOPL, which is computationally inadequate – several models expressed in different formalisms are necessary in most meaningful domains. Note also that any plan containing actions that are to be executed by a robot should also be translated to actionable metric terms that are understandable by the robot. For instance, specific positions in which objects are to be placed, locations to navigate to, and action dispatch times should be indicated in the plan. Finally, the knowledge represented in different models a robot should use for reasoning (the causal model being only one of these) often presents nontrivial interdependences: the capacity of the robot's tray (resource model) determines the number of trips a robot has to perform to and from the table it is clearing (causal reasoning), the type of meal (ontological model) may affect the spatial layout of the cutlery (spatial reasoning), and the time it takes for a beverage to get cold (temporal model) will affect the order of goal achievement (causal reasoning). The development of problem solving algorithms that account for these interdependences is an active topic of research. An overview of recent results in this direction is provided in Sect. 14.3.4.

Of particular interest to this discussion are temporal and spatial KR formalisms. Several temporal and spatial logics possess properties that make them both epistemological and computationally adequate for a variety of robotic problems. Below, we outline the principal temporal and spatial KR formalisms that are relevant to robotics.

14.3.2 Reasoning About Time

Linear temporal logic (LTL) [14.28] is a decidable propositional logic which provides a means to describe how execution paths will evolve in the future. The temporal operators $\bigcirc$ (next), $\Box$ (always), $\Diamond$ (eventually), $\mathcal{U}$ (until) and $\mathcal{R}$ (release) are used to represent conditions on the state of a system. For instance, *the robot will eventually place the cup on the table* (reachability), *the robot will never serve a cold coffee* (safety), and *the robot will always reach its charging station before its battery is discharged* (liveness). A fragment of LTL (specifically, syntactically co-safe LTL formulae [14.29]) has become relevant in robotics because of its ability to predicate upon discrete temporal properties of motions. For example, an exploration robot may be required to visit locations A, B and C in a predetermined sequential order; or conditions like *avoid C unless A or B have been visited, and always avoid D*. Given such temporal goal specifications, it is possible to compute motions that satisfy these goals with probabilistic roadmaps (PRMs) [14.30, 31].

LTL formulae express qualitative temporal specifications, thus offering the domain expert a means to specify conditions on execution at a high level of abstraction. LTL does not however consider that robot actions have a temporal extent, and that the temporal extents of different actions may be required to have specific relations. The qualitative temporal logics point algebra (PA) [14.32] and interval algebra (IA) [14.33] can be used to capture (and enforce) such relations. Specifically, they allow to represent qualitative temporal relations among temporal variables. Variables in PA represent time points, which can be used to represent events, such as *the robot has reached the table*, or start/end times of actions, such as *the robot starts to place a mug on the table*. In IA, variables represent time intervals, e.g., *the robot navigates to the table*, or *the robot places the mug on the table*. The basic PA relations are the three temporal relations $\{<, >, =\}$. All

Avidbots Ex1016
Page 381 of 2154

unions of these relations ($\leq$, $\geq$, $\neq$, the universal relation $\top$, and the empty relation $\emptyset$) are also valid PA constraints. In IA, constraints represent temporal relations among temporal intervals. The basic IA relations B_{IA} are the 13 possible temporal relations between intervals, namely *precedes* (p), *meets* (m), *overlaps* (o), *during* (d), *starts* (s), *finishes* (f), their inverses (e.g., p^{-1}), and *equals* ($\equiv$) (Fig. 14.5). A constraint in IA is a disjunction of basic relations $\{r_1, \ldots, r_n\} \in B_{\mathrm{IA}} \times \ldots \times B_{\mathrm{IA}}$. Going back to our example, a relevant piece of temporal knowledge for our waiter robot is that the action of picking the mug temporally overlaps the fact that it is holding the mug, i. e., the IA relation

$$\text{Pick(Mug)}\ \{\mathrm{o}\}\ \text{Holding(Mug)}\ . \tag{14.8}$$

This knowledge in fact represents the qualitative state of affairs in the presence of a successful pick-up action, whereas the contingency in which the mug slips from the robot's gripper (an all too common occurrence with today's robots!) can be expressed by the temporal relation

$$\text{Pick(Mug)}\ \{\mathrm{d}^{-1}\}\ \text{Holding(Mug)}\ . \tag{14.9}$$

A set of variables and constraints in PA or IA constitutes a constraint satisfaction problem (CSP) [14.34]. A fundamental reasoning problem in temporal CSPs is to test the *satisfiability* of the CSP – assessing whether a substitution of temporal values to variables exists that satisfies all constraints. A related problem is that of computing the *minimal* representation of a given CSP, that is, the equivalent CSP formed by the strongest implied constraints. Both problems are tractable for PA and for several fragments of IA [14.35]. A particularly useful example of tractable subalgebra of IA is the set of *convex* IA relations [14.36]. For example, {b, m, o} is a convex IA relation, whereas {b, o} is not. These good computational properties can be leveraged by robots during plan execution. Ensuring robust execution in the case of our waiter robot picking a mug need not be hard-coded as an ad-hoc status checking procedure into the implementation of the pick up action. Conversely, it can be modeled as a simple relation as above, and its verification can be cast as a CSP containing the relations

$$\begin{aligned}&\text{Pick(Mug)}\ \{\mathrm{o}\}\ \text{Holding(Mug)}\ ,\\ &\text{Pick(Mug)}\ \{\mathrm{d}^{-1}\}\ \text{Holding(Mug)}\ .\end{aligned} \tag{14.10}$$

The first constraint represents the requirement we expect to hold in nominal execution, whereas the second constraint results from perception and proprioception (more on this later). The above CSP is not satisfiable: the observed and executed situations are not consistent with the modeled temporal relation representing successful plan execution. This determination can be made in polynomial time (with respect to the number of variables in the CSP) with a particular form of inference, called *path consistency* [14.37].

Although convenient for specifying high-level requirements on robot plans, PA and IA cannot capture metric information. For this, we need metric temporal constraints, which allow to predicate on durations and relative temporal distances between variables. An important metric temporal problem is the temporal constraint satisfaction problem (TCSP). As in PA, variables represent time points. Constraints represent disjunctions of bounds on the temporal distance between the pairs of time points: given two time points a and b, the constraint

$$a \xrightarrow{[l_1,u_1]\vee[l_2,u_2]\vee\ldots[l_n,u_n]} b$$

states that $l_i \leq b - a \leq u_i$, for one or more $i \in \{1 \ldots n\}$. A TCSP is satisfiable iff at least one disjunct (pair of

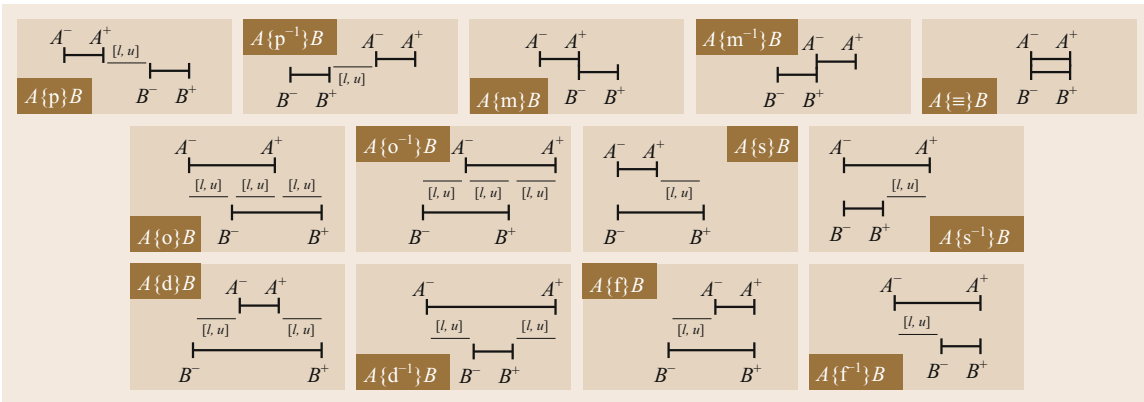


Fig. 14.5 The 13 basic qualitative relations in interval algebra. $[l, u]$ indicate metric bounds that can be attached to the basic relations to refine the metric semantics of the relations

Avidbots Ex1016
Page 382 of 2154

inequalities) per constraint holds. Unlike PA and IA, variables in a TCSP are associated with an explicit set of possible assignments of time values, called *domain*. Computing the minimal representation consists of restricting the domains of time points to the smallest intervals of time containing values that satisfy all constraints. A particular restriction of the TCSP, called the simple temporal problem (STP), has the property that domains are contiguous intervals of time. The condition for remaining in the STP fragment of TCSP is that constraints have only one disjunct, i. e., are of the form

$$a \xrightarrow{[l,u]} b \,.$$

Whereas computational problems for the the general TCSP are NP-hard, the satisfiability and minimal representation problems for the STP are tractable. Both problems are solved by path consistency inference [14.38], and highly optimized algorithms have been developed for the satisfiability and minimal representation problems [14.39, 40].

The semantics of basic PA and IA can be understood through the use of metric temporal constraints. For instance, the constraint

$$\text{Pick(Mug)}\ \{\text{o}\}\ \text{Holding(Mug)} \tag{14.11}$$

is equivalent to the metric constraints

$$\begin{aligned} &\text{Pick(Mug)}^- \xrightarrow{[1,\infty)} \text{Holding(Mug)}^- \,, \\ &\text{Holding(Mug)}^- \xrightarrow{[1,\infty)} \text{Pick(Mug)}^+ \,, \end{aligned} \tag{14.12}$$

where $(\cdot)^-$ and $(\cdot)^+$ represent, respectively, the start and end times of the corresponding intervals. Moreover, metric bounds can be used to define a metric extension of IA in which we can express high-level relations between intervals with additional metric bounds (Fig. 14.5). For instance, $A\{\text{p}[5, 13]\}B$ states that interval A should end at least 5 and at most 13 time units before interval B starts. In addition to binary constraints, it is also common practice to define the two unary constraints Release$[l, u]A$ and Deadline$[l, u]A$, stating, respectively, that A starts between l and u time units after the origin of time, and that A ends between l and u time units after the origin of time.

Overall, constraint-based temporal calculi can represent both qualitative and metric time. Provided that appropriate fragments of IA are used in modeling, the reasoning tasks can be conveniently reduced to path consistency inference in an STP. This is an attractive feature for robot reasoning: it provides a means to anchor observations and proprioception in time. In the example above, the robot must infer from sensing that the relation

$$\text{Pick(Mug)}\ \{\text{d}^{-1}\}\ \text{Holding(Mug)} \tag{14.13}$$

holds. An implementation of this capability can be achieved by representing as time points both observed and planned behavior, and constraining these time points so as to reflect the precise times in which they were observed. Assuming the current time is 20, that the robot started to pick the mug at time 5, and that the gripper reports that it is holding an object between times 10 and 18, the following STP models the robot's situation

$$\begin{aligned} &\text{Pick(Mug)}\ \{\text{o}[1,\infty)[1,\infty)[1\infty)\}\ \text{Holding(Mug)}\,, \\ &\text{Release}[5,5]\ \text{Pick(Mug)}\,, \\ &\text{Deadline}[20,20]\ \text{Pick(Mug)}\,, \\ &\text{Release}[10,10]\ \text{Holding(Mug)}\,, \\ &\text{Deadline}[18,18]\ \text{Holding(Mug)}\,. \end{aligned} \tag{14.14}$$

The above STP is not satisfiable, reflecting the fact that execution has not gone according to plan. Again, this state of affairs can be ascertained in low-order polynomial time, hence providing a simple mechanism for online fault detection.

The use of temporal constraint reasoning in planning and plan execution goes beyond diagnosing failures. The fact that intervals are grounded on a metric temporal CSP (an STP) provides a means to maintain knowledge about when actions should be dispatched, and whether they should be delayed to accommodate unexpected contingencies. For instance, we can express that the robot should start moving back to the counter once the mug has been held for at least 3 s with the constraint

$$\begin{aligned} &\text{Holding(Mug)}\ \{\text{o}[3,\infty)[1,\infty)[1,\infty)\} \\ &\quad \text{Move(Table, Counter)}\,. \end{aligned} \tag{14.15}$$

A delay in the start time of the Holding(Mug) predicate will *propagate* to the start time of the Move(Table, Counter) action, progressively pushing its start time into the future until the robot starts holding the mug. Correct propagation of exogenous events is ensured by maintaining in the STP the requirements of the plan together with constraints representing when events are observed, when actions are dispatched, and when they terminate. A simple procedure for maintaining these constraints is the following:

- When an action is dispatched at time t, a Release$[t, t]$ constraint is imposed on the corresponding interval, constraining its start time to the current time.

Avidbots Ex1016
Page 383 of 2154

- The fact that the action is still executing at time $t+m$ is modeled with the constraint Deadline$[t+m+1, \infty)$, modeling the fact that the action will end some time in the future.
- When the lower level executive signals at time $t+m+n$ that an action has finished, the constraint Deadline$[t+m+n, t+m+n]$ is added to the STP, thus constraining the action's end time.

Every time a constraint is added, the minimal STP is recomputed – that is, the domains of the time points (start and end times of actions and predicates) are updated. In so doing, we ensure that the lower bound of time points representing the start times of actions will always correspond to the earliest admissible time at which actions should be dispatched to the robot's executive layer. This guarantees that execution will uphold all temporal constraints modeled in the temporal CSP. These may include additional requirements on the plan, such as specifications regarding necessary orderings of actions, actions that must occur concurrently, and synchronizations with exogenous events.

The computational adequacy of temporal CSPs entails that reasoning about temporal models can be performed online during execution. Thus, reasoning about time is a way to enable fault diagnosis, guarantee timely action dispatching, and enforce temporal specifications on robot plans. Mechanisms based on temporal constraint reasoning have been used as a tool for this purpose in many settings involving execution of plans on real physical systems [14.41, 42]. Similar techniques have been used to account for uncontrollable events in plan execution [14.43], inferring context from sensor traces [14.44], and integrated context inference and planning [14.45]. A good overview of the fundamental principles of temporal constraint reasoning underlying these results is given in Dechter's book on constraint processing [14.46].

14.3.3 Reasoning About Space

Spatial KR formalisms serve the purpose of specifying desired spatial relations in a scene. As for temporal models, they can be used to make sense of sensor traces, to enforce conditions during plan execution, as well as to drive the planning process itself. Most work has focused on the use of qualitative spatial relations for scene understanding (e.g., in the context of perceptual anchoring [14.47]). Structural pattern recognition, important applications of which can be found in medical domains, employ cognitive vision techniques to match qualitative spatial knowledge (representing a specified structure) to perceived context [14.48, 49]. In many applications, qualitative relations do not belong to a well-defined calculus, rather are tailored to capture specific features (e.g., distance, orientation, and shape) which are useful for pattern specification and recognition in particular applications. As is the case for temporal knowledge, a well-defined spatial calculus is useful because of its provable formal properties – e.g., tractability of specific reasoning problems, expressiveness of the language, and so on. Well-founded spatial calculi permit logical reasoning, and, not surprisingly, they are grounded on similar principles as PA and IA. We illustrate here these principles and show simple examples of their use in robotic applications.

The main entities of interest in qualitative spatial calculi are objects, or, rather, the regions (or points) of physical space that they occupy. Spatial calculi provide a means to represent the relations among these regions. There exist several well-known and well-studied qualitative calculi. Each of these calculi focuses on one category of spatial concepts – e.g., topology, direction and distance. Region connection calculus (RCC) [14.50] is used for representing and reasoning with topological relations. Cardinal direction calculus (CDC) [14.51] is an approach based on directional relations. As for temporal calculi, these calculi use the language of constraints to represent spatial properties, and classical constraint-based reasoning techniques like path consistency [14.52] to ascertain consistency.

Of particular interest here is RCC, which is grounded on eight spatial relations describing the connectedness of regions: *disconnected* (DC), *externally connected* (EC), *tangential proper part* (TPP), *nontangential proper part* (NTPP), *partially overlapping* (PO), *equal* ($\equiv$), and the inverses TPP^{-1} and NTPP^{-1} (Fig. 14.6).

The full algebra deriving from the eight basic RCC relations is called RCC-8. An important restriction of RCC-8 is RCC-5, obtained by subsuming the relations DC and EC into one relation, and NTPP and TPP into another. The satisfiability and minimal CSP problems are NP-hard for both RCC-8 and RCC-5 – however, these problems are tractable if we restrict the language to the basic relations (as for IA), and a large number of tractable fragments of RCC-8 and RCC-5 are known [14.53]. A good example of where these properties are useful is in the problem of anchoring modeled relations to observed spatial layouts. This is exemplified in the work by *Cohn* et al. [14.54] on video sequence analysis, where the problem is to construct qualitative representations of the evolution of spatial relations in video sequences. The qualitative descriptions are obtained online, due to the computational adequacy of the qualitative spatial calculus used (a derivative of RCC called CORE-9).

Qualitative spatial reasoning can be used, much like qualitative temporal reasoning, to robustify plan execu-

Avidbots Ex1016
Page 384 of 2154

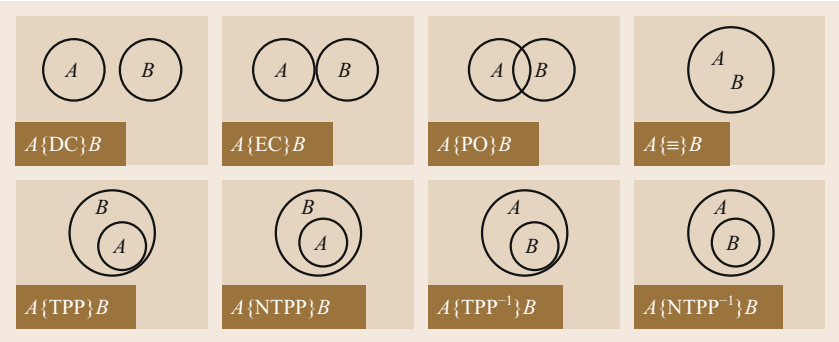


Fig. 14.6 The eight basic qualitative relations in RCC

tion. We may, for instance, require that mugs should be placed *on top of* saucers (Mug NTPP Saucer). As we have shown for temporal reasoning, the task of verifying the correct state of affairs can be cast to a CSP which is guaranteed to be feasible iff the desired spatial layout is observed.

However, note that all we can express with RCC constraints are topological relations between objects. RCC cannot express notions like *left-of*, *above*, and so on, as the only assumption made on variables is that they are convex regions of space. Conversely, rectangle algebra (RA) [14.55] expresses more than topological relations – at the cost of assuming that objects have a particular shape. Specifically, RA is an extension of IA to two dimensions. Its variables thus represent axis-parallel rectangles, and relations are pairs of IA relations, one for each axis.

RA can be augmented, like IA, with metric bounds. The resulting calculus, called ARA$^+$ [14.56], subsumes both topology and cardinal relations. For instance (Fig. 14.7), the relation

$$B\ \langle \mathrm{p}[5, 13], \mathrm{p}\rangle\ A \tag{14.16}$$

states that B precedes A in both the x and y axes, and that the distance along the x axis between A and B should be at least 5 and at most 13. The ARA$^+$ relation subsumes the qualitative relation *A is Northeast of B*, as well as the RCC relation $A\{\mathrm{DC}\}B$. Note that rectangular regions are compatible with the representation of segmented objects in most off-the-shelf perception modules. This property facilitates the process of going from sensor data to symbols. Toward this aim, ARA$^+$ provides the unary constraints $\mathrm{At}[l_1, u_1][l_2, u_2][l_3, u_3][l_4, u_4]$ and $\mathrm{Size}[l_1, u_1][l_2, u_2][l_3, u_3][l_4, u_4]$. The first constraint bounds the length of the sides of a rectangle, while the second bounds the placement of a rectangle in 2-D space. Note that the At constraint performs a similar function to the Release and Deadline constraints used in the metric extension of IA, which denote specified or perceived absolute placements in time of intervals. It is thus intuitive to see how the At constraint can be *implemented* with Release and Deadline constraints in the augmented IA CSPs used to represent the projections of rectangles on the two axes.

We can employ ARA$^+$ constraints to represent desired placements of objects, both in qualitative and metric terms. For instance, the specification of a well-set table for our waiter robot could be

$$\begin{aligned}
&\mathrm{Fork}\ \langle \mathrm{d}[5, +\infty)[5, +\infty),\\
&\qquad \mathrm{d}[5, +\infty)[5, +\infty)\rangle\ \mathrm{Table}\ ,\\
&\mathrm{Knife}\ \langle \mathrm{d}[5, +\infty)[5, +\infty),\\
&\qquad \mathrm{d}[5, +\infty)[5, +\infty)\rangle\ \mathrm{Table}\ ,\\
&\mathrm{Fork}\ \langle \mathrm{p}, \mathrm{d}\rangle\ \mathrm{Mug}\ ,\\
&\mathrm{Mug}\ \langle \mathrm{p}, \mathrm{d}\rangle\ \mathrm{Knife}\ ,\\
&\mathrm{Mug}\ \mathrm{Size}[8, 8][8, 8]\ ,\\
&\mathrm{Fork}\ \mathrm{Size}[2, 2][15, 15]\ ,\\
&\mathrm{Knife}\ \mathrm{Size}[2, 2][15, 15]\ ,
\end{aligned} \tag{14.17}$$

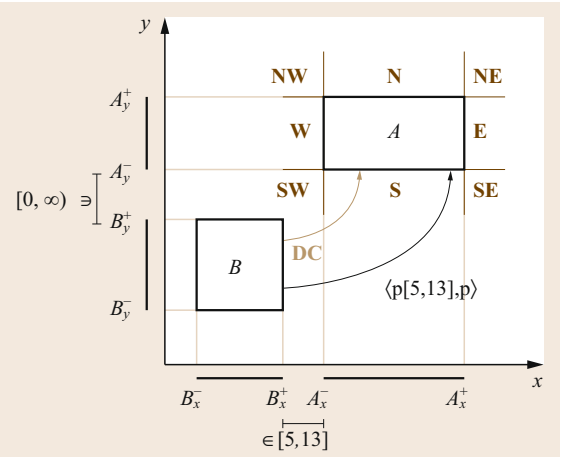


Fig. 14.7 The ARA$^+$ relation $B\langle \mathrm{p}[5, 13], \mathrm{p}\rangle A$, which subsumes RCC relation $A\{\mathrm{DC}\}B$ and CDC relation A Northeast of B

Avidbots Ex1016
Page 385 of 2154

that is, forks and knives should be at least 5 cm from the edge of the table, forks should be located on the left of mugs and knives on the right, the size of forks and knives is $2 \times 15\,\text{cm}^2$, and the size of mugs is $8 \times 8\,\text{cm}^2$.

The above relations can be maintained, much like we have illustrated for temporal reasoning, in a lower-level metric constraint-based representation – in this case, consisting of two STPs, one for each axis of the reference frame. It has been shown [14.56] that these two STPs are satisfiable iff the ARA$^+$ CSP is satisfiable, and that the minimal STPs contain all the admissible placements of objects. The minimal STPs can thus be used to extract an admissible placement of the objects in the scene. Technically, this is done by *hallucinating* the objects that are to be placed in the scene, that is, representing both observed objects and those that are intended to be placed into the scene in the spatial CSP. In the example above, supposing the robot must place the mug on a table which already has a fork and a knife on it, the hallucinated object is the mug, while the fork and knife would be constrained by At constraints representing their observed locations. The bounds of all variables in the minimal STPs represent all the admissible placements of the objects. Those of the fork and knife will not have been refined in the minimal representation, as they were already fixed by the At constraints resulting from observation; conversely, the bounds of the Mug rectangle are refined in the minimal representation to contain only those points that are admissible wrt. the other constraints in the spatial CSP.

So far, we have shown how employing constraint-based temporal and spatial KR formalisms allows to cast the problem of fault detection to a tractable problem that can be solved online. Another important aspect of plan execution that is facilitated by these representations is fault identification and repair [14.57]. It is easy to see why an explicit representation formulated in terms of constraints facilitates these tasks: suppose that the fork and knife are placed on the table correctly from the qualitative point of view – i. e., the fork is left of the knife – but that they are only 5 cm apart. The resulting spatial CSP would not be satisfiable, as it would be impossible to fit the mug (whose bounding box is $8 \times 8\,\text{cm}^2$) between the two pieces of cutlery, as prescribed by constraints (Fork$\langle$p, d$\rangle$Mug) and (Mug$\langle$p, d$\rangle$Knife). The spatial CSP can be used at this point as a tool to perform *culprit analysis*: the same CSP without the At constraint modeling the observed placement of the fork represents the situation in which we assume that the fork can be moved; if this CSP is satisfiable, then we know that replacing the fork is a way to achieve a well-set table.

Spatial reasoning has not received as much attention as temporal reasoning in the context of robotic applications. The AI and Robotics research communities are, however, quickly clustering around this important topic. Spatial calculi are a well studied topic in AI, albeit with less applications in robotics than temporal reasoning. As robots become more competent in manipulation and navigation tasks, the issue of representing and enforcing high-level requirements on space begins to come to the fore. As always, it is the combination of epistemological and computational adequacy of significant fragments of spatial KR formalisms that makes them useful in robot applications. Indeed, the potential of qualitative and metric spatial reasoning goes beyond planning and plan execution, with active research directions in human–robot interaction [14.58–60] and object search [14.61]

14.3.4 Reasoning About Multiple KR Formalisms

Despite their simplicity, the examples we have used to illustrate the basic concepts of temporal and spatial reasoning point to the fact that reasoning about action, time, space, and any other knowledge possessed by the robot must occur jointly. For instance, the decision to place the mug on the table may depend on the spatial layout of other objects on the table, or on whether there is a more urgent task to be carried out before doing so. It is easy to see that even this simple task cannot ignore kinematics and geometric constraints, which may call for approaching the table from a different side, or to move another object before placing the mug. Recent work in integrated task and motion planning is a response to the need for integrating these forms of reasoning into planning. Today, this is very much seen as the next big challenge, and consistent research results in this specific area of hybrid reasoning are closing the gap between classical planning and real robotic applications [14.62–67].

We argue, however, that kinematics and geometric constraints are only two of the important types of knowledge necessary for achieving competent robot behavior. Comparatively less work exists in integrating other types of knowledge. DL reasoing has been employed to generate high-level plans [14.68], to refine planning domains [14.69], and *Galindo* et al. [14.70] show how to enhance the task planning process with spatial and ontological models. The largest volume of work in integrating diverse forms of reasoning into planning focuses on integrating planning, temporal, and resource reasoning – e.g., planning with metric temporal reasoning [14.71], qualitative temporal models [14.44], combined qualitative and metric temporal

reasoning [14.72], and resource scheduling [14.73–75].

Although these advancements contribute to widening the expressive power of models for robot reasoning, they do not suggest general methods for jointly reasoning about heterogeneous KR formalisms. As noted earlier, the types of knowledge a robot should possess ultimately depend on the application, which makes this an important basic research question. A few examples pointing in the direction of general solutions do exist, although work in the area is sparse. These include planning modulo theories [14.76], an extension of satisfiabiliy modulo theories (SMTs) [14.77], which enrich classical planning models with semantics belonging to arbitrary first order theories. SMTs themselves are an example of general hybridization scheme, and have been employed in a robotic context by *Nedunuri* et al. [14.78] to enforce metric requirements on motion and manipulation tasks. The approach known in the literature as meta-constraint reasoning is similar in structure to SMTs, but is grounded on the more general notion of CSP. Meta-constraint reasoning has been used to develop solvers for highly expressive planning domains that include qualitative and metric time, resources, and logical constraints [14.79]. In the context of robotic applications, the approach has been used for online planning with qualitative and metric time, spatial relations in ARA$^+$ and resources [14.57], as well as for online configuration planning for multirobot systems [14.80].

14.3.5 Knowledge Representation Systems for Robots

In autonomous robot control systems, the kinds of information that we have discussed in the previous subsections are not only represented and reasoned about but also used subsymbolically for generating perception-guided activity. For example, some information such as the appearance and poses of the objects on the table might be generated by the perception system of the robot through the interpretation of sensor data. Other information such as the position of the robot in the environment might be estimated using state estimation algorithms and stored in dedicated data structures. Also, symbolic action descriptions have to be translated into low-level parameterizations of control routines in order to generate behavior and cause effects.

These aspects of embodying abstract and symbolic reasoning in robots are at least partly addressed by system-level research on KR and reasoning for robots. The resulting systems serve as integrated question-answering components in a robot's control program that gather information from different sources and employ a range of inference methods for controlling and parameterizing control routines.

Examples of recent systems include the pntology-based unified robot knowledge (OUR-K) framework [14.81], mostly used for navigation actions, the ORO system [14.82] that focuses on knowledge for human–robot interaction and dialog grounding, and the KnowRob system [14.83] providing deep knowledge for robotic manipulation tasks. The semantic layer [14.84] of the PEIS Ecology [14.85] serves both autonomous robots and ambient intelligent environments. In the Ke Jia project [14.86], Answer Set Programming is used as representation for integrating KR, natural language processing, task planning and motion planning on a mobile platform. Also classical cognitive architectures such as Soar [14.87] and ACT-R also integrate modules for storing knowledge [14.88] with inference capabilities, but have only rarely been used on robots [14.89].

One field that has seen much progress in the past years is the creation and representation of environment models, often termed *semantic maps*. Advances in object recognition allowed to detect and perceptually interpret single objects in environments, which enables robots to store higher level semantic information about them in their respective maps. The term *semantic* is used for a range of different interpretation capabilities – from a mere segmentation and categorization of the environment into different region types to the segmentation of objects, their categorization, and the interpretation of their properties [14.90], or from the co-occurrence of objects in scenes [14.91] up to environment representations in description logics [14.92], from statistical relational environment models [14.93] to an embedding of spatial information into an encyclopedic and common-sense knowledge base [14.94]. Many of the reasoning tasks that we have discussed in Sect. 14.3.3 have implicitly assumed expressive semantic maps that can make symbolic place descriptions such as *on the table* or *in the cupboard* effective.

Knowledge representation aspects also become increasingly important for the instruction of robots with natural language as well as for human–robot interaction. For understanding directives involving objects in the environment [14.95] and for following spoken directions [14.96–99], a robot might have to ground the words it hears or reads into its percepts and actions. The natural-language text may be the result of direct interaction with a human, but can also be obtained from resources on the Web, for example, for mining object locality knowledge [14.100] or extracting task instructions from websites [14.101] (VIDEO 705).

Web-enabled robotics is another branch of robotics that has recently become increasingly important. Modern robot control systems use the worldwide web as an information resource [14.102], for sharing knowledge and skills [14.103], for telepresence applications [14.104], to employ web services [14.105], and to use computational resources provided through the web [14.106] (VIDEO 706).

14.4 Plan-Based Robot Control

Plan-based control is an information processing approach for controlling autonomous robots that aims at making robot control systems more general – in at least three respects. First, it enables robots to successfully carry out multiple, diverse, and possibly interfering tasks. Second, it can improve the course of action by adapting the control software for the respective execution context. Third, plan-based control can enable robots to perform novel tasks without requiring the collection of extensive experience.

In the plan-based approach, robots generate actions and behavior by synthesizing, maintaining, and executing plans, where plans are robot control programs that a robot cannot only execute but also reason about and manipulate [14.107]. Thus plan-based controllers can manage and adapt plans in order to better achieve complex and changing tasks [14.108]. The use of plans enables robots to flexibly interleave complex and interacting tasks, exploit opportunities, quickly plan their courses of action, and, if necessary, revise their intended activities.

Plans in plan-based control have two roles. They are (1) executable prescriptions that can be interpreted by the robot to generate behavior in order to accomplish its jobs and (2) representations that can be synthesized and revised by the robot to meet the robot's criterion of utility. Besides having means for representing and generating plans, plan-based controllers are also equipped with tools that enable computational processes to (1) predict what might happen when a robot controller gets executed and return the result as an execution scenario, (2) infer what might be wrong with a robot controller given an execution scenario, and (3) perform complex revisions on robot controllers.

In most general terms robot planning can be considered to be the automatic generation, refinement, revision, and optimization of robot plans [14.107]. As a computational problem it can be formulated as follows: given an abstract plan P and a description of the current situation, find an executable realization Q that maximizes some objective function V. In this problem formulation, an abstract plan might be to go shopping and to clean up the apartment, to win a robot soccer game, to monitor the traffic in a particular area using an autonomous helicopter, or, for a museum tour-guide robot, to inform and entertain the visitors of the museum.

Robot planning algorithms aim at finding appropriate plans by hypothesizing possible courses of action and predicting what could happen when the plans get executed. Based on these predictions, the robot decides whether the plan will meet its objectives.

In the remainder of this subsection we will first discuss conceptual models underlying robot planning and the representation of robot plans. We will then talk about the automated synthesis of robot plans and finally discuss mechanisms for revising plans and optimizing plans based on experience.

14.4.1 Models of Plan-Based Control

The generation of plans and the prediction of what will happen is based on models of the robot, its actions, and the world. The most commonly used models are state-transition systems (also called discrete-event systems or problem spaces) [14.27].

Using state-transition systems, the behavior and the effects of robot activity can be represented as a triple $\Sigma = \langle S, A, \gamma \rangle$, where:

- $S = \{s_1, s_2, \ldots\}$ is a set of states
- $A = \{a_1, a_2, \ldots\}$ is a set of actions
- $\gamma: S \times A \rightarrow S$ is a state-transition function.

Thus, the action repertoire of a robot can be stated as a graph where the nodes represent the states of the environment and links of the form

$$s_i \xrightarrow{A} s_j$$

represent the fact that the state s_i can be transformed into the state s_j by the robot executing action A.

Imagine a robot has to stack three different objects, e.g., a cup, a saucer, and a set. In the initial state, all objects are placed on a table surface. The robot can pick an object (provided that there is no other object on top of it) and can place it either on the surface or on top of another object (provided, again, that there is nothing already on top of it). If we further assume that each object can be placed on all other objects, and that only a single

Avidbots Ex1016
Page 388 of 2154

object can be moved at a time (and not a stack of two of them), then we get the state space depicted in Fig. 14.8.

The state-transition model abstracts away aspects of robot activity such as the structure of actions and the events, behavior, and effects that are caused during action execution. Actions are considered to cause atomic state transitions.

Even with these abstractions, a number of variations of planning problems are investigated. These variations result from different trade-offs one can make with respect to the computational complexity of the planning problem, the strength of assumptions one makes about the capabilities of the robot, and the realism with which one wants to represent the behavior of the robot and the effects of its actions.

An important subclass of state-transition models are the ones that characterize action by specifying their preconditions and effects [14.25, 109, 110]. Preconditions state the conditions under which the respective action is executable and has the specified effects. The effects represent how the world changes when the action is executed.

If these models are stated as axioms in a logical representation language (Sect. 14.2) planning methods can be particularly powerful: they can compute sequences or partially ordered sets of actions that are provably working plans from the logical point of view. This means that planning algorithms can find plans that are guaranteed to transform any state satisfying a given state description into a state that satisfies a given goal, if such a plan exists. Unfortunately, in many robot applications action models that are axiomatized this way are too unrealistic.

Many extensions of the state-transition model deal with providing the expressive power needed to state action models more realistically. The first one extends the state-transition model by allowing state transitions to be caused by a combination of an action and an event ($\gamma: S \times A \times E \rightarrow S$) rather than the action alone. This modification allows us to represent action failures by stating that a failure event occurs while the action is executed. It also allows us to approximate environments that are dynamically changing. The second extension is to change the state transition function such that it maps into a subset of states rather than a single state ($\gamma: S \times A \rightarrow 2^S$). This modification allows the robot to reason through the nondeterminism that can often not be avoided when autonomous robots perform realis-

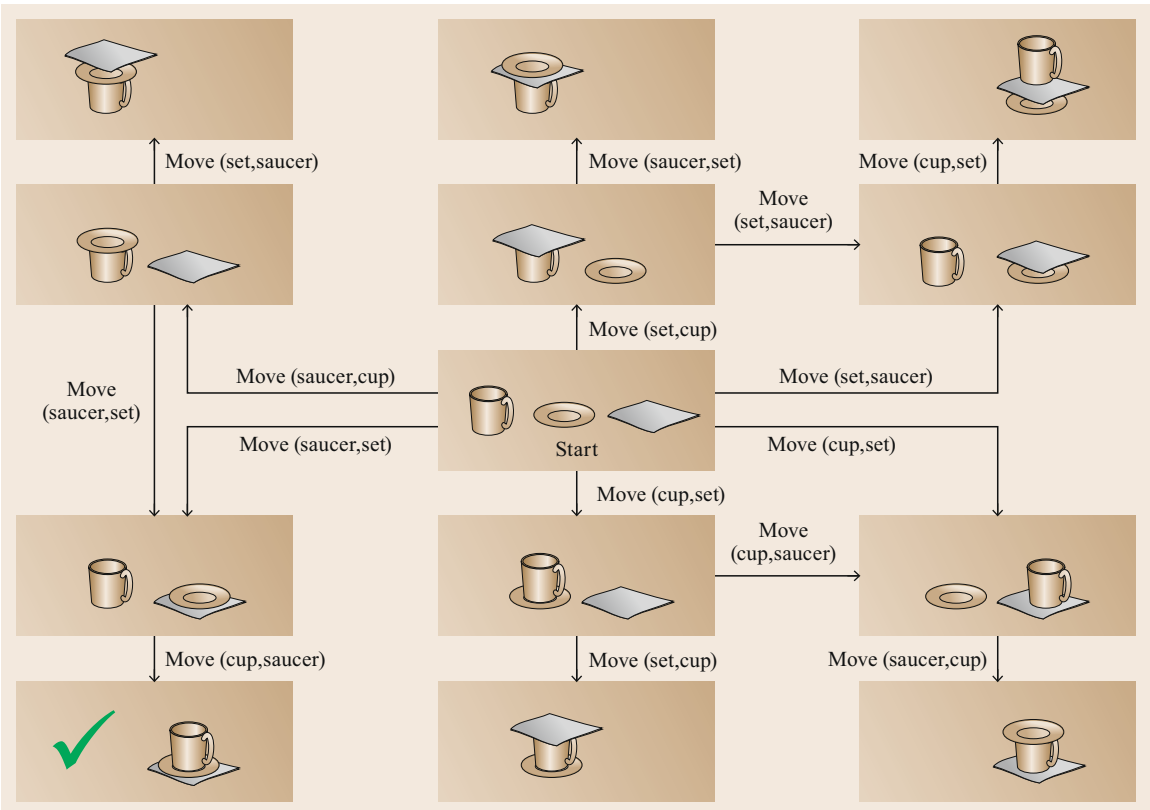


Fig. 14.8 Example: problem space for stacking three objects

Avidbots Ex1016
Page 389 of 2154

tic activities. Other researchers extend models used for plan-based control by including aspects that allow the robot to reason about its resources, geometric aspects of manipulation actions [14.24, 111], sensing actions, and so on.

Besides the state-transition model researchers also use hybrid system models to reason about robot activities [14.112–114]. In the hybrid system model the action repertoires of robots are specified as continuous variable dynamical systems with a phased operation. Within each phase, called control mode, the system evolves continuously according to the dynamical law of that mode, called continuous flow. Thus, the state of the hybrid system can be thought of as a pair – the control mode and the continuous state. The control mode identifies a flow, and the continuous flow identifies a position in it. Also associated with each control mode are so-called jump conditions, specifying the conditions that the discrete state and the continuous state together must satisfy to enable a transition to another control mode. The transitions can cause abrupt changes of the discrete as well as the continuous state. The jump relation specifies the valid settings of the system variables that might occur during a jump. Then, until the next transition, the continuous state evolves according to the flow identified by the new control mode.

The advantage of a hybrid-system-based conceptualization over state-based ones is that hybrid systems can adequately model concurrent processes with interfering continuous effects. They also allow for discrete changes in process parameterization, which we need to model the activation, deactivation, and reparameterization of control processes through concurrent reactive plans (Sect. 14.4.2). In addition, hybrid system based conceptualizations can model the procedural meaning of waiting for and reacting to asynchronous events.

While the increased expressive power allows us to model the behavior that today's autonomous manipulation robots generate much more faithfully, the models are also in most cases too complex to completely synthesize action plans from primitive actions. Many of the modelling issues to be addressed in the intersection between action planning and control are discussed in [14.115, 116].

14.4.2 Representation of Plans

When the robot is performing its activity, it is under the control of a plan. The most common forms of action plans in autonomous robot control are:

- State-transition plans, totally or partially ordered sets of atomic actions.
- Policies, mappings from (perceived) environment states to actions.
- Reactive plans that specify how the robot is to respond to sensory events and execution failures in order to accomplish its jobs.

State-Transition Plans

Robot action plans that are generated according to the state-transition model are ordered sets of actions, often called plan steps. The assumptions of the state-transition model, in particular that plan steps are atomic and their execution can be abstracted away, are not satisfied in many robot applications. To execute plan steps successfully, actions have to include subactions such as detecting the object to be fetched in a scene or detecting that the object has been successfully grasped before lifting the arm. As autonomous robot activity is far from perfect, the robot might not be able to generate the necessary execution events or it can generate the events but fails to achieve the desired effects. For example, an object might slip out of the robot's hand after grasping it.

As a consequence, state-transition plans can often only serve as guidelines for generating robust and flexible robot behavior [14.117]. To close the gap to low-level robot control, researchers have proposed multilayer software architectures for autonomous robot activity, in particular 3T (3-tiered) architectures [14.118] (Chap. 12). 3T architectures run planning and execution at different software layers and different time scales where a sequencing layer synchronizes between both layers. Each layer uses a different form of plan or behavior specification language. The planning layer typically uses a problem space plan, whereas the execution layer employs feedback control routines that can be activated and deactivated. The intermediate layer typically uses a simple form of a reactive plan language (see below). Even though 3T architectures have been shown to produce competent activity in some application domains, there remain important issues to be addressed for more complex applications. One of them is that the behavior generated by the sequencing layer might deviate substantially from the action model used for planning, which might cause plans not to work as expected by the action models used by the planning component. As the planning layer abstracts away from control structures, it also lacks the expressiveness to specify robust and flexible behavior.

MDP Policies

A second category of plans are policies, which are mappings from discretized robot states into robot actions. Such policies are computed as solutions of Markov decision problems (Sect. 14.2.2). The policies computed

Avidbots Ex1016
Page 390 of 2154

by MDPs aim at robustness and optimizing the average performance. They are less appropriate when the consequences of decisions are not in the near future. The decision of how to grasp an object might considerably affect the ease with which the object can be placed later in the activity episode. Extensions of MDP policies such as the combination of programming and reinforcement learning [14.119, 120] and the structuring of policies with so-called options [14.121, 122] partly address these issues.

Concurrent Reactive Plan Languages

Concurrent reactive plan languages are the result of applying programming language design methods to the specification of robot behavior. Concurrent reactive plans specify how the robot is to respond to events, including perception and failure events, in order to achieve its goals [14.123]. Such plan languages offer rich sets of control structures that support the structuring and transparent synchronization and monitoring of concurrent control processes, the detection and recovery from execution failures, the context-dependent execution of subplans, the prompt reaction to asynchronous events, and data structures for the descriptions of objects to be manipulated [14.124, 125]. Some reactive plan languages also incorporate explicit representations of the beliefs, goals, and intentions of the robot [14.126–128] and even provide powerful reasoning mechanisms to adapt the plan execution to the respective situation context [14.129].

While concurrent reactive plan languages enable robots to accomplish their tasks competently, they are too expressive for automatically generating robust, flexible high-performance plans from first principles. They typically require the use of plan libraries and transformational planning techniques to deal with the computational complexity caused by the expressiveness of the plan languages (Sect. 14.4.4).

14.4.3 Generating Robot Action Plans

The most common form of reasoning performed on plans is the synthesis of plans for given problems. In general, planning is concerned with reasoning about intended courses of action before executing them in order to avoid failures in achieving given goals, to prevent undesired side effects, or to improve the performance of the robot in accomplishing its goals.

There are many different variants of planning problems, categories of plans that can be generated, and methods that can be applied to planning problems. Comprehensive overviews can be found in textbooks [14.4, 27] and overview articles [14.130, 131]. A wide range of the planning problems is addressed within an international AI planning competition (IPC) [14.132] using standardized problem specification languages [14.109, 110]. Several implementations of planning algorithms participating in this competition are available as open-source software.

In planning for state-transition, the task of the planning system is to find an ordered set of actions that transform any state satisfying the initial state description into a state satisfying the given goals.

The first class of planning systems are systems that enumerate the states that can be reached by prefixes (or suffixes) of hypothetical action sequences until a goal state is reached. Planning systems that fall into this category include the STRIPS planning algorithm [14.25] and newer and more efficient algorithms such as the FF (fast forward) planning algorithm [14.133] as well as various variants of these algorithms.

Another class of algorithms enumerates abstract plans that represent sets of plans. The idea underlying this approach is that by reasoning about an abstract plan, the planning algorithm can compactly reason about different plan instances at the same time by reasoning about their common causal structure. For example, a partial order plan represents all possible completions of the plan. The POP (partial-order planning) algorithms is a family of algorithms that is primarily used for teaching action planning [14.130] while GraphPlan [14.134] is an algorithm that is designed for planning process efficiency [14.131].

The algorithms above are domain independent: they compute plans by analyzing whether preconditions of plan steps are achieved through the effects of prior actions using general algorithms. In contrast, domain-dependent planners use domain knowledge to constrain and prune the search space to be explored in a planning process. An example of such domain knowledge is that stacks of objects are built bottom up. Domain-dependent planners are often much more efficient than their domain-independent counterparts. Examples of domain-dependent planners are the TL- (temporal logic) [14.135] and the TAL planners [14.19].

Another category of planners that are designed to deal with more complex planning problems are HTN (hierarchical task network) planners. HTN planners are equipped with libraries of plan schemata that specify how abstract plan steps can be decomposed into more complex subplans. Examples of such planners are the SHOP planners [14.136] and extensions thereof that are combined with description logics and applied to autonomous robot control [14.69].

The planning approaches above generate plans that have similar expressiveness in terms of the behavior they can specify. The planners mainly differ with re-

Avidbots Ex1016
Page 391 of 2154

spect to the size of the planning problems they can handle and their efficiency in solving planning problems.

14.4.4 Revising and Improving Plans

Another dimension that robot action planning methods can be scaled along is the expressiveness of the plans they generate in terms of specifying competent robot behavior. A subclass of planning methods that is suitable for generating concurrent reactive plans is *transformational planning*.

The underlying idea of transformational planning is to paste together plans from libraries of plan schemata and revise them for the specified task [14.137] rather than creating completely new plans from scratch. Transformational planning has also been referred to as *plan repair* or *plan debugging* [14.138, 139] and is often performed within a *generate-test-debug* control strategy [14.140]. Plan transformation is a promising approach in applications where default plans that have a substantial chance of being successful can be constructed easily, tested by projecting them, analyzed with respect to their expected execution failures, and where failures can be used to index the appropriate plan revisions.

Transformational planning subsumes classical planning methods such as partial-order planning in the sense that the operations of the POP algorithms can be simply expressed as transformations. In addition to the transformations that correspond to classical planning, transformational planning typically employs collections of transformations including domain-specific ones, heuristic transformations, and transformations for plan optimization [14.141]. In cases where the planners employ transformations that cannot be guaranteed to work, they rely on powerful plan projection methods for testing the generated plan hypotheses [14.142].

Transformational planning is a promising approach for autonomous robots performing complex manipulation tasks because it makes it easy for programmers to specify how plans should be modified to avoid certain expected behavior flaws. For example, a transformation could tell the robot to stack items in order to transport them more efficiently subject to additional constraints such as that it should not stack too many items because it might drop them. Or, it might stack one kind of object on another one but not vice versa – typically it is safe to stack cups on plates but not plates on cups.

Transformational planning can also effectively facilitate the learning not only of plans [14.139] but also of planning capabilities by learning for a particular robot and environment what the robot can stack and carry and what not.

14.4.5 Other Reasoning Methods

There are a number of other reasoning methods that plan-based robot control needs to employ in order to produce competent object manipulation capabilities [14.143]. One of the main reasons is that the tasks and subtasks that a robot is to perform are typically underspecified: the robot might be asked to fetch the screwdriver from the table but when it gets there are several different screwdrivers. Or, the robot might be asked to crack an egg and has to decide where to crack it, where to put the yolk and egg white, what to do with the shell, etc. When fetching objects, it has to reason about abstract states such as visibility and reachability of objects. We can only give a couple of examples here.

The first reasoning problem is that of *object identity resolution*, the inference task of deciding whether two internal representations refer to the same object in the real world. When receiving the command to pick up the red cup from the table, the robot must make at some point the decision that a data structure returned by the robot's perception system and the language description refer to the same object. Only by making this inference, the robot is able to infer that it can achieve the given task by picking up a particular object that it sees. In logic-based representations this problem is even harder because symbols used in the planning problem specification *denote* objects in the outside world. In this setting the problem of object identity resolution problem becomes the *symbol grounding problem* [14.1], i. e., the problem to determine *the right* or, at least, *the most likely* object in a given environment for each constant and each definite description in the logical representation. The most powerful inference methods to reason about object identity resolution are first-order probabilistic reasoning mechanisms as described in [14.144].

Other important inference methods are needed to model some predicates used in the activity models more realistically. Predicates that deserve such special treatment include *visibility* and *reachability*. For example, the Asymov planner [14.111] reasons about action preconditions that include reachability conditions and tests reachability through the existence of motion plans. Other planning systems reason about the next best views that a robot should take in order to perform complex perception tasks. Sophisticated action planners that include action and motion planning employing more general geometric reasoning capabilities can be found in [14.66]. Cognition-enabled plans employ such reasoning methods in order to parameterize actions with respect to the effects that should be achieved and avoided [14.62].

Avidbots Ex1016
Page 392 of 2154

14.5 Conclusions and Further Reading

The aim of this chapter was, first, to motivate the purpose and benefit of employing general reasoning methods in robot control; second, to sketch some of the generic reasoning methods from AI that are well-understood, available, and most likely of use on a robot; and third, to go into plan-based robot control in some more detail, as it is a particularly obvious form of employing reasoning to the end of improving robot action.

Yet, the chapter has only sketched the tip of an iceberg. Employing symbolic reasoning in an autonomous mobile robot requires to have and update on the robot a symbolic model of the world, or at least of that part of the world which is relevant for the current task – but who knows beforehand exactly what is and what isn't relevant for some task? It therefore requires to be able to interpret the current sensor data stream on a semantic level to transform it into a symbolic description in terms of categories that are meaningful in the KR. Assuming that it is not practical to equip a robot with all knowledge that may likely be relevant for each and every situation it may find itself in, it requires some form of learning, generalization, and/or the ability to look up unknown knowledge from available sources. These and some more requirements that are needed for practically using AI reasoning methods on a robot cannot be fulfilled completely based on current state-of-the-art AI methods and technology; as a consequence, currently existing robot prototypes employing AI reasoning methods in their control systems are limited in terms of generality and adaptability.

It is no recent insight that employing AI reasoning methods on robots is both a chance and a challenge. In fact some of the early influential work in AI was driven by the idea of designing integrated *reasoning robots*, such as SHAKEY [14.145] (VIDEO 704); as a consequence, early AI textbooks like [14.146] would naturally include robotics as a target field for using AI reasoning methods. The field of AI has since focused on applications other than Robotics. Recent efforts in integrating reasoning systems with autonomous robots have done much to bring the original robot-inspired vision back into focus, as witnessed by series of workshops on *AI-Based Robotics* (e.g., IROS 2013) or *AI and Robotics* (AAAI 2014). It is exactly the family of challenges and chances that this chapter is about that is discussed at these events.

14.5.1 Further Reading

For a more complete coverage of the generic methods described here, as well as the rest of AI, we refer to AI textbooks, of which the one by *Russell* and *Norvig* [14.4] is comprehensive and currently most widely used. Among the sources for first-hand material, let us mention the two journals *Artificial Intelligence* and *Journal of Artificial Intelligence Research* (JAIR) [14.147], which both cover well the topics addressed here. The main international conference in the field is IJCAI (*International Joint Conference on Artificial Intelligence*); ECAI (*European Conference on Artificial Intelligence* [14.148]) and AAAI (*Association for the Advancement of Artificial Intelligence Conference on Artificial Intelligence* [14.149]) are other major international symposia on Artificial Intelligence in general. Regarding planning in particular, ICAPS (*International Conference on Automated Planning and Scheduling* [14.150]) is the main international conference.

Reference [14.151] reviews about the same topic as this chapter, but structures it differently along the notion of *deliberation functions*.

Video-References

VIDEO 704 SHAKEY: Experimentation in robot learning and planning (1969)
available from http://handbookofrobotics.org/view-chapter/14/videodetails/704

VIDEO 705 From knowledge grounding to dialogue processing
available from http://handbookofrobotics.org/view-chapter/14/videodetails/705

VIDEO 706 RoboEarth final demonstrator
available from http://handbookofrobotics.org/view-chapter/14/videodetails/706

Avidbots Ex1016
Page 393 of 2154

References

14.1 S. Harnad: The symbol grounding problem, Physica D **42**, 335–346 (1990)

14.2 S. Coradeschi, A. Saffiotti: An introduction to the anchoring problem, Robotics Auton. Syst. **43**(2/3), 85–96 (2003)

14.3 F. Baader, D. Calvanese, D. McGuinness, D. Nardi, P. Patel-Schneider (Eds.): *The Description Logic Handbook* (Cambridge Univ. Press, Cambridge 2003)

14.4 S.J. Russell, P. Norvig: *Artificial Intelligence: A Modern Approach*, 3rd edn. (Pearson Education, Upper Saddle River 2010)

14.5 R.J. Brachman, H.J. Levesque: *Knowledge Representation and Reasoning* (Morgan Kaufmann, San Francisco 2004)

14.6 W.O. van Quine: *Methods of Logic*, 4th edn. (Harvard Univ. Press, Cambridge 1982)

14.7 Z. Manna, R. Waldinger: *The Deductive Foundations of Computer Programming: A One-Volume Version of "The Logical Basis for Computer Programming"* (Addison-Wesley, Reading 1993)

14.8 W. Hodges: Elementary predicate logic. In: *Handbook of Philosophical Logic*, Vol. 1, ed. by D. Gabbay, F. Guenthner (D. Reidel, Dordrecht 1983)

14.9 A. Robinson, A. Voronkov (Eds.): *Handbook of Automated Reasoning* (Elsevier, Amsterdam 2001)

14.10 M. Davis, G. Logemann, D. Loveland: A machine program for theorem proving, Communications ACM **5**(7), 394–397 (1962)

14.11 The international SAT Competitions web page: http://www.satcompetition.org/

14.12 The Web Ontology Language OWL: http://www.w3.org/TR/owl-features/

14.13 OWL 2 Web Ontology Language Document Overview (Second Edition): http://www.w3.org/TR/owl2-overview/

14.14 P. Hitzler, M. Krötzsch, S. Rudolph: *Foundations of Semantic Web Technologies* (Chapman Hall/CRC, Boca Raton 2009)

14.15 J. McCarthy, P. Hayes: Some philosophical problems from the standpoint of artificial intelligence, Mach. Intell. **4**, 463–507 (1969)

14.16 H. Levesque, R. Reiter, Y. Lespérance, F. Lin, R. Scherl: Golog: A logic programming language for dynamic domains, J. Log. Program. **31**, 59–83 (1997)

14.17 M. Shanahan, M. Witkowski: High-level robot control through logic, ATAL '00: Proc. 7th Int. Workshop Intell. Agents VII. Agent Theor. Archit. Lang. (2001) pp. 104–121

14.18 M. Thielscher: *Reasoning Robots. The Art and Science of Programming Robotic Agents* (Springer, Berlin 2005)

14.19 P. Doherty, J. Kvarnström: TALplanner: A temporal logic based planner, AI Magazine **22**(3), 95–102 (2001)

14.20 K.L. Chung, F. AitSahila: *Elementary Probability Theory*, 4th edn. (Springer, Berlin, Heidelberg 2003)

14.21 J. Pearl: *Probabilistic Reasoning in Intelligent Systems* (Morgan Kaufmann, San Mateo 1988)

14.22 A.R. Cassandra, L.P. Kaelbling, M.L. Littman: *Acting Optimally in Partially Observable Stochastic Domains*, Tech. Rep. AAAI-94 (Department of Computer Science, Brown University, Providence 1994) pp. 1023–1028

14.23 R.E. Bellman: *Dynamic Programming* (Princeton Univ. Press, Princeton 1957)

14.24 L.P. Kaelbling, T. Lozano-Pérez: Integrated task and motion planning in belief space, Int. J. Robotics Res. **32**(9/10), 1194–1227 (2013)

14.25 R.E. Fikes, N.J. Nilsson: Strips: A new approach to theorem proving in problem solving, J. Artif. Intell. **2**, 189–208 (1971)

14.26 P.E. Hart, N.J. Nilsson, B. Raphael: A formal basis for the heuristic determination of minimum cost paths, IEEE Trans. Syst. Sci. Cybern. **4**(2), 100–107 (1968)

14.27 M. Ghallab, D.S. Nau, P. Traverso: *Automated Planning – Theory and Practice* (Elsevier, Amsterdam 2004)

14.28 A. Pnueli: The temporal logic of programs, Proc. 18th Annu. Symp. Found. Comput. Sci., Providence (1977) pp. 46–57

14.29 O. Kupferman, M.Y. Vardi: Model checking of safety properties, Form. Methods Syst. Des. **19**(3), 291–314 (2001)

14.30 E. Plaku, L.E. Kavraki, M.Y. Vardi: Hybrid systems: From verification to falsification by combining motion planning and discrete search, Form. Methods Syst. Des. **34**, 157–182 (2009)

14.31 A. Bhatia, M.R. Maly, L.E. Kavraki, M.Y. Vardi: Motion planning with complex goals, IEEE Robotics Autom. Mag. **18**(3), 55–64 (2011)

14.32 M. Vilain, H. Kautz, P. van Beek: Constraint propagation algorithms for temporal reasoning: A revised report. In: *Readings in Qualitative Reasoning About Physical Systems*, ed. by D.S. Weld, J. de Kleer (Morgan Kaufmann, San Francisco 1990) pp. 373–381

14.33 J. Allen: Towards a general theory of action and time, Artif. Intell. **23**(2), 123–154 (1984)

14.34 E.P.K. Tsang: *Foundations of Constraint Satisfaction* (Academic Press, London, San Diego 1993)

14.35 P. Jonsson, A. Krokhin: Complexity classification in qualitative temporal constraint reasoning, Artif. Intell. **160**(1/2), 35–51 (2004)

14.36 G. Ligozat: A new proof of tractability for ORD-horn relations, AAAI Workshop Spat. Temp. Reason., Portland (1996)

14.37 U. Montanari: Networks of constraints: Fundamental properties and applications to picture processing, Inf. Sci. **7**, 95–132 (1974)

14.38 R.W. Floyd: Algorithm 97: Shortest path, Communications ACM **5**(6), 345 (1962)

14.39 L. Xu, B.Y. Choueiry: A new efficient algorithm for solving the simple temporal problem, Proc. 4th Int. Conf. Temp. Log., Cairns (2003)

Avidbots Ex1016
Page 394 of 2154

14.40 L.R. Planken, M.M. De Weerdt, R.P.J. van der Krogt: P3C: A new algorithm for the simple temporal problem, Proc. Int. Conf. Autom. Plan. Sched. (ICAPS), Sydney (2008) pp. 256–263

14.41 C. McGann, F. Py, K. Rajan, J. Ryan, R. Henthorn: Adaptive control for autonomous underwater vehicles, Proc. 23rd Natl. Conf. Artif. Intell., Chicago (2008) pp. 1319–1324

14.42 B.C. Williams, M.D. Ingham, S.H. Chung, P.H. Elliott: Model-based programming of intelligent embedded systems and robotic space explorers, Proceedings IEEE **91**(1), 212–237 (2003)

14.43 T. Vidal, H. Fargier: Handling contingency in temporal constraint networks: From consistency to controllabilities, J. Exp. Theor. Artif. Intell. **11**, 23–45 (1999)

14.44 P. Doherty, J. Kvarnström, F. Heintz: A temporal logic-based planning and execution monitoring framework for unmanned aircraft systems, J. Auton. Agents Multi-Agent Syst. **2**(2), 332–377 (2009)

14.45 F. Pecora, M. Cirillo, F. Dell'Osa, J. Ullberg, A. Saffiotti: A constraint-based approach for proactive, context-aware human support, J. Ambient Intell. Smart Environ. **4**(4), 347–367 (2012)

14.46 R. Dechter: *Constraint Processing*, The Morgan Kaufmann Series in Artificial Intelligence (Morgan Kaufmann, San Francisco 2003) pp. 155–165

14.47 A. Loutfi, S. Coradeschi, M. Daoutis, J. Melchert: Using knowledge representation for perceptual anchoring in a robotic system, Int. J. Artif. Intell. Tools **17**(5), 925–944 (2008)

14.48 O. Colliot, O. Camara, I. Bloch: Integration of fuzzy spatial relations in deformable models – Application to brain MRI segmentation, Pattern Recogn. **39**(8), 1401–1414 (2006)

14.49 X. Wang, J.M. Keller, P. Gader: Using spatial relationships as features in object recognition, Annu. Meet. North Am. Fuzzy Inf. Proces. Soc., Syracuse (1997)

14.50 D.A. Randell, Z. Cui, A.G. Cohn: A Spatial Logic based on Regions and Connection, Proc. Int. Conf. Princ. Knowl. Represent. Reason., Cambridge (1992)

14.51 S. Skiadopoulos, M. Koubarakis: Composing cardinal direction relations, Artif. Intell. **152**(2), 143–171 (2004)

14.52 J. Renz, B. Nebel: Qualitative spatial reasoning using constraint calculi. In: *Handbook of Spatial Logics*, ed. by M. Aiello, I. Pratt-Hartmann, J.F.A.K. van Benthem (Springer, Berlin, Heidelberg 2007) pp. 161–215

14.53 T. Drakengren, P. Jonsson: A complete classification of tractability in Allen's algebra relative to subsets of basic relations, Artif. Intell. **106**(2), 205–219 (1998)

14.54 A.G. Cohn, J. Renz, M. Sridhar: Thinking inside the box: A comprehensive spatial representation for video analysis, Proc. 13th Int. Conf. Princ. Knowl. Represent. Reason., Rome (2012) pp. 588–592

14.55 P. Balbiani, J.-F. Condotta, L. Farinas Del Cerro: A new tractable subclass of the rectangle algebra, Proc. 16th Int. Jt. Conf. Artif. Intell., Stockholm (1999) pp. 442–447

14.56 M. Mansouri, F. Pecora: A representation for spatial reasoning in robotic planning, Proc. IROS Workshop AI-Based Robotics, Tokyo (2013)

14.57 M. Mansouri, F. Pecora: More knowledge on the table: Planning with space, time and resources for robots, IEEE Int. Conf. Robotics Autom. (ICRA), Hong Kong (2014) pp. 647–654

14.58 M. Skubic, D. Perzanowski, S. Blisard, A. Schultz, W. Adams, M. Bugajska, D. Brock: Spatial language for human-robot dialogs, IEEE Trans. Syst. Man Cybern. C **34**(2), 154–167 (2004)

14.59 R. Moratz, T. Tenbrink: Spatial reference in linguistic human-robot interaction: Iterative, empirically supported development of a model of projective relations, Spat. Cogn. Comput. **6**(1), 63–107 (2006)

14.60 S. Guadarrama, L. Riano, D. Golland, D. Gouhring, Y. Jia, D. Klein, P. Abbeel, T. Darrell: Grounding spatial relations for human-robot interaction, IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS), Tokyo (2013) pp. 1640–1647

14.61 L. Kunze, K.K. Doreswamy, N. Hawes: Using qualitative spatial relations for indirect object search, IEEE Int. Conf. Robotics Autom. (ICRA), Hong Kong (2014) pp. 163–168

14.62 L. Mosenlechner, M. Beetz: Parameterizing actions to have the appropriate effects, Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst., San Francisco (2011) pp. 4141–4147

14.63 A. Gaschler, R.P.A. Petrick, M. Giuliani, M. Rickert, A. Knoll: KVP: A knowledge of volumes approach to robot task planning, Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst., Tokyo (2013) pp. 202–208

14.64 G. Havur, K. Haspalamutgil, C. Palaz, E. Erdem, V. Patoglu: A case study on the Tower of Hanoi challenge: Representation, reasoning and execution, IEEE Int. Conf. Robotics Autom. (ICRA), Tokyo (2013) pp. 4552–4559

14.65 L. de Silva, A.K. Pandey, R. Alami: An interface for interleaved symbolic-geometric planning and backtracking, IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS), Tokyo (2013) pp. 232–239

14.66 L.P. Kaelbling, T. Lozano-Pérez: Hierarchical task and motion planning in the now, IEEE Int. Conf. Robotics Autom. (ICRA) (2011) pp. 1470–1477

14.67 F. Lagriffoul, D. Dimitrov, A. Saffiotti, L. Karlsson: Constraint propagation on interval bounds for dealing with geometric backtracking, Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst., Vilamoura (2012) pp. 957–964

14.68 G. de Giacomo, L. Iocchi, D. Nardi, R. Rosati: Moving a robot: the KR&R approach at work, Proc. 5th Int. Conf. Princ. Knowl. Represent. Reason. (KR'96), Cambridge (1996) pp. 198–209

14.69 R. Hartanto, J. Hertzberg: Fusing DL reasoning with HTN planning, Lect. Notes Comput. Sci. **5243**, 62–69 (2008)

14.70 C. Galindo, J.A. Fernandez-Madrigal, J. Gonzalez, A. Saffiotti: Using semantic information for improving efficiency of robot task planning, Proc.

Avidbots Ex1016
Page 395 of 2154

ICRA-07 Workshop Semant. Inf. Robotics, Rome (2007) pp. 27–32
14.71 W. Cushing, S. Kambhampati, Mausam, D.S. Weld: When is temporal planning really temporal?, Proc. 20th Int. Jt. Conf. Artif. Intell., Hyderabad (2007)
14.72 J.L. Bresina, A.K. Jónsson, P.H. Morris, K. Rajan: Activity planning for the Mars exploration rovers, Proc. 15th Int. Conf. Autom. Plan. Sched. (ICAPS), Monterey (2005) pp. 1852–1859
14.73 M. Cirillo, F. Pecora, H. Andreasson, T. Uras, S. Koenig: Integrated motion planning and coordination for industrial vehicles, Proc. 24th Int. Conf. Autom. Plan. Sched. (ICAPS), Portsmouth (2014)
14.74 S. Fratini, F. Pecora, A. Cesta: Unifying planning and scheduling as timelines in a component-based perspective, Arch. Control Sci. **18**(2), 231–271 (2008)
14.75 M. Ghallab, H. Laruelle: Representation and control in IxTeT, a temporal planner, Proc. 2nd Int. Conf. Artif. Intell. Plan. Syst. (AIPS-94), Chicago (1994) pp. 61–67
14.76 P. Gregory, D. Long, M. Fox, J.C. Beck: Planning modulo theories: Extending the planning paradigm, Proc. 15th Int. Conf. Autom. Plan. Sched. (ICAPS), São Paulo (2012)
14.77 R. Nieuwenhuis, A. Oliveras, C. Tinelli: Solving SAT and SAT modulo theories: From an abstract Davis–Putnam–Logemann–Loveland procedure to DPLL(T), Journal ACM **53**, 937–977 (2006)
14.78 S. Nedunuri, S. Prabhu, M. Moll, S. Chaudhuri, L.E. Kavraki: SMT-based synthesis of integrated task and motion plans for mobile manipulation, IEEE Int. Conf. Robotics Autom. (ICRA), Hong Kong (2014)
14.79 U. Köckemann, L. Karlsson, F. Pecora: Grandpa hates robots – Interaction constraints for planning in inhabited environments, Proc. 28th Conf. Artif. Intell., Quebéc City (2014)
14.80 M. Di Rocco, F. Pecora, A. Saffiotti: When robots are late: Configuration planning for multiple robots with dynamic goals, Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst., Tokyo (2013)
14.81 G.H. Lim, I.H. Suh, H. Suh: Ontology-based unified robot knowledge for service robots in indoor environments, IEEE Trans. Syst. Man Cybern. A **41**(3), 492–509 (2011)
14.82 S. Lemaignan, R. Ros, L. Mösenlechner, R. Alami, M. Beetz: ORO, a knowledge management module for cognitive architectures in robotics, Proc. 2010 IEEE/RSJ Int. Conf. Intell. Robots Syst., Taipei (2010) pp. 3548–3553
14.83 M. Tenorth, M. Beetz: KnowRob – A knowledge processing infrastructure for cognition-enabled robots, Int. J. Robotics Res. **32**(5), 566–590 (2013)
14.84 M. Daoutis, S. Coradeschi, A. Loutfi: Grounding commonsense knowledge in intelligent systems, J. Ambient Intell. Smart Environ. **1**(4), 311–321 (2009)
14.85 A. Saffiotti, M. Broxvall, M. Gritti, K. LeBlanc, R. Lundh, J. Rashid, B.S. Seo, Y.J. Cho: The PEIS-Ecology project: Vision and results, IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS), Nice (2008) pp. 2329–2335
14.86 X. Chen, J. Ji, J. Jiang, G. Jin, F. Wang, J. Xie: Developing high-level cognitive functions for service robots, Proc. 9th Int. Conf. Auton. Agents Multiagent Syst., Toronto (2010) pp. 989–996
14.87 J.F. Lehman, J.E. Laird, P. Rosenbloom: A gentle introduction to Soar, an architecture for human cognition, Invit. Cogn. Sci. **4**, 212–249 (1996)
14.88 N. Derbinsky, J.E. Laird: Extending soar with dissociated symbolic memories, Symp. Human Mem. Artif. Agents, AISB (2010) pp. 31–37, http://ai.eecs.umich.edu/soar/sitemaker/docs/pubs/aisb2010_rwwa_soar9.pdf
14.89 W.G. Kennedy, M. Rouleau, J.K. Bassett: Multiple levels of cognitive modeling within agent-based modeling, Proc. 18th Conf. Behav. Represent. Model. Simul., Sundance (2009) pp. 143–144
14.90 R.B. Rusu, Z.C. Marton, N. Blodow, M. Dolha, M. Beetz: Towards 3D point cloud based object maps for household environments, Robotics Auton. Syst. J. Semant. Knowl. Robotics **56**(11), 927–941 (2008)
14.91 S. Vasudevan, R. Siegwart: Bayesian space conceptualization and place classification for semantic maps in mobile robotics, Robotics Auton. Syst. **56**(6), 522–537 (2008)
14.92 H. Zender, O. Martinez Mozos, P. Jensfelt, G.J.M. Kruijff, W. Burgard: Conceptual spatial representations for indoor mobile robots, Robotics Auton. Syst. **56**(6), 493–502 (2008)
14.93 B. Limketkai, L. Liao, D. Fox: Relational object maps for mobile robots, Proc. Int. Jt. Conf. Artif. Intell. (IJCAI) (2005) pp. 1471–1476
14.94 M. Tenorth, L. Kunze, D. Jain, M. Beetz: KNOWROB-MAP – Knowledge-linked semantic object maps, 10th IEEE-RAS Int. Conf. Humanoid Robots, Nashville (2010) pp. 430–435
14.95 N. Mavridis, D. Roy: Grounded situation models for robots: Where words and percepts meet, Proc. 2006 IEEE/RSJ Int. Conf. Intell. Robots Syst., Beijing (2006) pp. 4690–4697
14.96 D.K. Misra, J. Sung, K. Lee, A. Saxena: Tell me Dave: Context-sensitive grounding of natural language to mobile manipulation instructions, Proc. Robotics Sci. Syst. (RSS) (2014)
14.97 T. Kollar, S. Tellex, D. Roy, N. Roy: Toward understanding natural language directions, Proc. 5th AMC/IEEE Int. Conf. Hum.-Robot Interact. (HRI), Osaka (2010) pp. 259–266
14.98 C. Matuszek, E. Herbst, L. Zettlemoyer, D. Fox: Learning to parse natural language commands to a robot control system, Proc. 13th Int. Symp. Exp. Robotics (ISER) Québec City (2012) pp. 403–415
14.99 F. Duvallet, T. Kollar, A. Stentz: Imitation learning for natural language direction following through unknown environments, 2013 IEEE Int. Conf. Robotics Autom. (ICRA) (2013) pp. 1047–1053
14.100 K. Zhou, M. Zillich, H. Zender, M. Vincze: Web mining driven object locality knowledge acquisition for efficient robot behavior, IEEE/RSJ Int.

Avidbots Ex1016
Page 396 of 2154

Conf. Intell. Robots Syst. (IROS), Vilamoura (2012) pp. 3962–3969
14.101 M. Tenorth, D. Nyga, M. Beetz: Understanding and executing instructions for everyday manipulation tasks from the World Wide Web, IEEE Int. Conf. Robotics Autom. (ICRA), Anchorage (2010) pp. 1486–1491
14.102 M. Tenorth, U. Klank, D. Pangercic, M. Beetz: Web-enabled robots – Robots that use the web as an information resource, Robotics Autom. Mag. **18**(2), 58–68 (2011)
14.103 M. Waibel, M. Beetz, R. D'Andrea, R. Janssen, M. Tenorth, J. Civera, J. Elfring, D. Gálvez-López, K. Häussermann, J.M.M. Montiel, A. Perzylo, B. Schießle, O. Zweigle, R. van de Molengraft: RoboEarth – A world wide web for robots, Robotics Autom. Mag. **18**(2), 69–82 (2011)
14.104 S. Osentoski, B. Pitzer, C. Crick, G. Jay, S. Dong, D.H. Grollman, H.B. Suay, O.C. Jenkins: Remote robotic laboratories for learning from demonstration – Enabling user interaction and shared experimentation, Int. J. Soc. Robotics **4**(4), 449–461 (2012)
14.105 M.B. Blake, S.L. Remy, Y. Wei, A.M. Howard: Robots on the web, IEEE Robotics Autom. Mag. **18**, 33–43 (2011)
14.106 D. Hunziker, M. Gajamohan, M. Waibel, R. D'Andrea: Rapyuta: The RoboEarth cloud engine, IEEE Int. Conf. Robotics Autom. (ICRA) (2013) pp. 438–444
14.107 D. McDermott: Robot planning, AI Magazine **13**(2), 55–79 (1992)
14.108 M.E. Pollack, J.F. Horty: There's more to life than making plans: Plan management in dynamic, multiagent environments, AI Magazine **20**(4), 71–83 (1999)
14.109 D. McDermott, M. Ghallab, A. Howe, C. Knoblock, A. Ram, M. Veloso, D. Weld, D. Wilkins: *PDDL – The Planning Domain Definition Language*, Tech. Rep. CVC TR-98-003/DCS TR-1165 (Yale Center for Computational Vision and Control, New Haven 1998)
14.110 M. Fox, D. Long: PDDL2.1: An extension of PDDL for expressing temporal planning domains, J. Artif. Intell. Res. **20**, 61–124 (2003)
14.111 F. Gravot, S. Cambon, R. Alami: aSyMov: A planner that deals with intricate symbolic and geometric problems, Springer Tracts Adv. Robotics **15**, 100–110 (2005)
14.112 R. Alur, T. Henzinger, H. Wong-Toi: Symbolic analysis of hybrid systems, Proc. 37th IEEE Conf. Decis. Control, Tampa (1997) pp. 702–707
14.113 R. Alur, T. Henzinger, P. Ho: Automatic symbolic verification of embedded systems, IEEE Trans. Softw. Eng. **22**, 181–201 (1996)
14.114 M. Beetz, H. Grosskreutz: Probabilistic hybrid action models for predicting concurrent percept-driven robot behavior, J. Artif. Intell. Res. **24**, 799–849 (2005)
14.115 K. Passino, P. Antsaklis: A system and control-theoretic perspective on artificial intelligence planning systems, Appl. Artif. Intell. **3**, 1–32 (1989)
14.116 T. Dean, M. Wellmann: *Planning and Control* (Morgan Kaufmann Publishers, San Mateo 1991)
14.117 R. Alami, R. Chatila, S. Fleury, M. Ghallab, F. Ingrand: An architecture for autonomy, Int. J. Robotics Res. **17**(4), 315–337 (1998)
14.118 R.P. Bonasso, R.J. Firby, E. Gat, D. Kortenkamp, D.P. Miller, M.G. Slack: Experiences with an architecture for intelligent, reactive agents, J. Exp. Theor. Artif. Intell. **9**(2/3), 237–256 (1997)
14.119 D. Andre, S. Russell: Programmable reinforcement learning agents, Proc. 13th Conf. Neural Inf. Process. Syst. (2001) pp. 1019–1025
14.120 D. Andre, S.J. Russell: State abstraction for programmable reinforcement learning agents, 18th Natl. Conf. Artif. Intell., Edmonton (2002) pp. 119–125
14.121 R.S. Sutton, D. Precup, S.P. Singh: Between MDPs and Semi-MDPs: A framework for temporal abstraction in reinforcement learning, Artif. Intell. **112**(1/2), 181–211 (1999)
14.122 D. Precup: Temporal Abstraction in Reinforcement Learning, Ph.D. Thesis (University of Massachusetts, Amherst 2000)
14.123 M. Beetz: Structured reactive controllers, J. Auton. Agents Multi-Agent Syst. **4**(1/2), 25–55 (2001)
14.124 D. McDermott: *A Reactive Plan Language* (Yale University, New Haven 1991)
14.125 M. Ingham, R. Ragno, B. Williams: A reactive model-based programming language for robotic space explorers, Proc. 6th Int. Symp. Artif. Intell. Robotics Autom. Space (ISAIRAS) (2001)
14.126 M. Bratman: *Intention, Plans, and Practical Reason* (Harvard Univ. Press, Cambridge 1987)
14.127 M. Bratman, D. Israel, M. Pollack: Plan and resource-bounded practical reasoning, Comput. Intell. **4**, 349–355 (1988)
14.128 M. Georgeff, F. Ingrand: Decision making in an embedded reasing system, Proc. 11th Int. Jt. Conf. Artif. Intell. (1989) pp. 972–978
14.129 M. Beetz, D. Jain, L. Mosenlechner, M. Tenorth, L. Kunze, N. Blodow, D. Pangercic: Cognition-enabled autonomous robot control for the realization of home chore task intelligence, Proceedings IEEE **100**(8), 2454–2471 (2012)
14.130 D.S. Weld: An introduction to least commitment planning, AI Magazine **15**(4), 27–61 (1994)
14.131 D.S. Weld: Recent advances in AI planning, AI Magazine **20**(2), 93–123 (1999)
14.132 D.V. McDermott: The 1998 AI planning systems competition, AI Magazine **21**(2), 35–55 (2000)
14.133 J. Hoffmann, B. Nebel: The FF planning system: Fast plan generation through heuristic search, J. Artif. Intell. Res. **14**, 253–302 (2001)
14.134 A.L. Blum, M.L. Furst: Fast planning through plan graph analysis, J. Artif. Intell. **90**, 281–300 (1997)
14.135 F. Bacchus, F. Kabanza: Planning for temporally extended goals, Ann. Math. Artif. Intell. **22**(1/2), 5–27 (1998)
14.136 D. Nau, O. Ilghami, U. Kuter, J.W. Murdock, D. Wu, F. Yaman: SHOP2: An HTN planning system, J. Artif. Intell. Res. **20**, 379–404 (2003)

Avidbots Ex1016
Page 397 of 2154

14.137 D. McDermott: *Transformational planning of reactive behavior*, Tech. Rep. (Yale University, New Haven 1992)
14.138 P. H. Winston: *Learning Structural Descriptions from Examples*, AI Tech. Rep. 231 (MIT, Cambridge 1970)
14.139 K.J. Hammond: *Case-Based Planning: Viewing Planning as a Memory Task* (Academic Press, Waltham 1989)
14.140 R.G. Simmons: A theory of debugging plans and interpretations, Proc. 7th Natl. Conf. Artif. Intell. (1988) pp. 94–99
14.141 M. Beetz: *Concurrent Reactive Plans: Anticipating and Forestalling Execution Failures*, Lecture Notes in Artificial Intelligence, Vol. 1772 (Springer, Berlin, Heidelberg, 2000)
14.142 H. Grosskreutz: Probabilistic projection and belief update in the pGOLOG framework. In: *Informatik 2000*, Informatik Aktuell, ed. by K. Mehlhorn, G. Snelting (Springer, Berlin, Heidelberg 2000) pp. 233–249
14.143 L. Morgenstern: Mid-sized axiomatizations of commonsense problems: A case study in egg cracking, Studia Log. **67**(3), 333–384 (2001)
14.144 N. Blodow, D. Jain, Z.-C. Marton, M. Beetz: Perception and probabilistic anchoring for dynamic world state logging, Proc. 10th IEEE-RAS Int. Conf. Humanoid Robots (Humanoids) (2010) pp. 160–166
14.145 N.J. Nilsson: *Shakey the Robot*, Tech. Note , Vol. TN 323 (SRI International, Stanford 1984) http://www.ai.sri.com/shakey/
14.146 B. Raphael: *The Thinking Computer: Mind Inside Matter* (W.H. Freeman, San Francisco 1976)
14.147 Journal of Artificial Intelligence Research: http://www.jair.org/
14.148 European Conference on Artificial Intelligence: http://www.eccai.org/ecai.shtml
14.149 AAAI Conference on Artificial Intelligence: http://www.aaai.org/Conferences/AAAI/aaai.php
14.150 International Conference on Automated Planning and Scheduling: http://www.icaps-conference.org/
14.151 F. Ingrand, M. Ghallab: Robotics and artificial intelligence: A perspective on deliberation functions, AI Communications **27**(1), 63–80 (2014)

Avidbots Ex1016
Page 398 of 2154

Multimedia Contents

15. Robot Learning

Jan Peters, Daniel D. Lee, Jens Kober, Duy Nguyen-Tuong, J. Andrew Bagnell, Stefan Schaal

Machine learning offers to robotics a framework and set of tools for the design of sophisticated and hard-to-engineer behaviors; conversely, the challenges of robotic problems provide both inspiration, impact, and validation for developments in robot learning. The relationship between disciplines has sufficient promise to be likened to that between physics and mathematics. In this chapter, we attempt to strengthen the links between the two research communities by providing a survey of work in robot learning for learning control and behavior generation in robots. We highlight both key challenges in robot learning as well as notable successes. We discuss how contributions tamed the complexity of the domain and study the role of algorithms, representations, and prior knowledge in achieving these successes. As a result, a particular focus of our chapter lies on model learning for control and robot reinforcement learning. We demonstrate how machine learning approaches may be profitably applied, and we note throughout open questions and the tremendous potential for future research.

15.1 **What Is Robot Learning** 358
15.1.1 Robot Learning Approaches 358
15.1.2 Robot Learning Settings 359
15.1.3 Notation 360
15.2 **Model Learning** 360
15.2.1 Model Types 361
15.2.2 Model Learning Architectures 363
15.2.3 Model Learning Challenges and Constraints 365
15.2.4 Model Learning Methods 366
15.2.5 Applications of Model Learning 369
15.2.6 Future Directions 371
15.3 **Reinforcement Learning** 372
15.3.1 Relation to Classical Control 372
15.3.2 Reinforcement Learning Methods 373
15.3.3 Optimal Control with Learnt Models 375
15.3.4 Model-Free Value Function Approaches 376
15.3.5 Policy Search 377
15.3.6 Challenges in Robot Reinforcement Learning 380
15.3.7 Tractability Through Representation 382
15.3.8 Tractability Through Prior Knowledge 383
15.3.9 Tractability Through Models 384
15.4 **Conclusions** 385
Video-References 386
References 386

In the future, robots will be part of daily life in human society, providing assistance in many areas ranging from clinical applications, education and care giving, to normal household environments, disaster scenarios, and space exploration [15.1]. Robots will no longer be used to only execute the same task thousands of times, but rather they will be faced with thousands of different tasks that rarely repeat in an ever changing environment [15.2]. It will be difficult to preprogram all possible tasks and scenarios in such robots, and robots will need to be able to learn by themselves or with the help of humans. They will need to automatically adjust to stochastic and dynamic environments, as well as to changes due to wear and tear. Machine learning will be necessary to achieve this high degree of autonomy and enabling future robots to perceive and act in the world.

Avidbots Ex1016
Page 399 of 2154

15.1 What Is Robot Learning

Robot learning consists of a multitude of machine learning approaches in the context of robotics. The type of the learning problem is usually characterized by the type of feedback, the process of data generation, and the type of data. At the same time, the type of data will determine the robot learning approach which actually can be employed. While the field of robot learning includes a huge number of topics ranging from learning in perception, state abstraction, decision making, and probabilistic techniques in robotics [15.3–5], this chapter focuses on issues of learning action generation and control. As indicated in Fig. 15.1, we explain robot learning using the two key ingredients: (i) the setting in terms of data and feedback, and (ii) the learning approach.

15.1.1 Robot Learning Approaches

In this chapter, we will focus on the core robot learning approaches capable of learning action generation and control. Particularly, we will be focusing on techniques of *model learning* and *reinforcement learning* (RL). (Note that we omit imitation learning by behavioral cloning as Chap. 75 deals with it extensively.) These methods have enabled real-time learning in complex robot systems such as helicopters, legged robots, anthropomorphic arms, and humanoid robots. We first discuss some broader issues of learning control before reviewing technical details and the related literature.

Learning Models for Control

Models are among the most essential tools in robotics, and it is widely believed that intelligent mammals also rely on internal models in order to generate their actions. A model describes essential predictive information about the behavior of the environment and the influence of an agent on this environment. While classical robotics relies on manually generated models based on human insights into physics, future autonomous

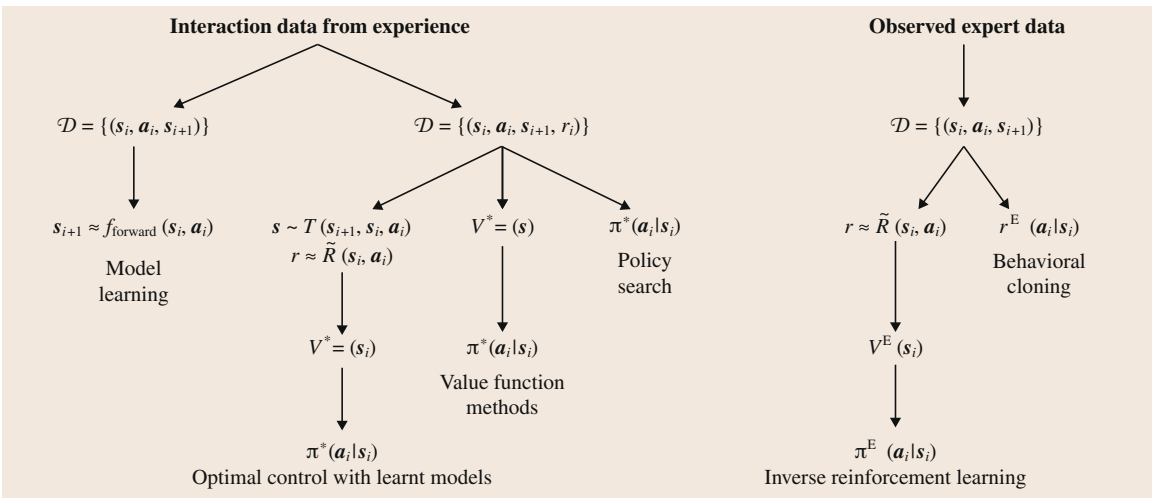


Fig. 15.1 The main branches of robot learning for action generation and control can be characterized both in terms of the available data $\mathcal{D}$ and the employed approach that learns from the data. A key distinction for data sets is whether they were generated from observed expert's demonstration or the robot's embodied experience. Data sets generically include samples i that consist at least of states s_i, actions a_i, and next states s_{i+1}. However, in many cases, a quality assessment in form of a reward r_i is available as feedback. The branches in the tree illustrate the core robot learning problems considered here. Model learning generate a model $T(s_{i+1}, s_i, a_i)$ from data. Reinforcement learning approaches require a reward in order to obtain an optimal policy π^*. Model-based methods usually learn a model from which they subsequently construct an optimal value function and policy. Model-free methods either learn value functions from data without model knowledge to construct a policy while policy search methods optimize the policy directly based on the interaction data. Expert data allows for reproducing observed behavior π^{E}. Two branches have had a particular importance in robot learning: imitation learning by behavioral cloning can treated by supervised learning and is discussed in detail in Chap. 74. Imitation learning by inverse RL reconstructs the reward function of the expert and uses reinforcement learning approaches to re-construct the expert's policy (This figure was inspired by [15.6])

Avidbots Ex1016
Page 400 of 2154

robots need to be able to automatically generate models from information which is extracted from the data streams accessible to the robot. Machine learning approaches in generating these models will also need to be adapted to the control architectures for which they are intended. Such model learning for control will be discussed in Sect. 15.2.

Planning with Reinforcement Learning

Planning motor behaviors and optimizing them with respect to some criteria is part of every motor skill. Reinforcement learning offers one of the most general frameworks to address these issues. In essence, it assumes that the goal or intent of a movement is represented in terms of an optimization or reward criterion, such that learning algorithms can fill in the often nontrivial details of the plan. While RL has wide area of application in many domains ranging from computer games to elevator scheduling [15.7], its application to high-dimensional physical systems with continuous state–action spaces requires often dedicated approaches, algorithms, representations, and prior knowledge. An important issue will also be the choice between model-based and model-free algorithms, as well as between value function-based and policy search methods. Reinforcement learning's different branches are indicated in Fig. 15.1 and will be discussed in Sect. 15.3.

Learning from Demonstration with Inverse Reinforcement Learning

Learning a skill starting from scratch with a complex robot system is often infeasible or extremely time consuming. Taking inspiration from child teaching techniques, learning from demonstration has become a very popular tool to endow a robot with some initial reasonable performance, from which further learning can progress more rapidly. As another chapter in this handbook will address learning from demonstration in more detail, we also discuss a particular subclass of learning from demonstration, i. e., using inverse RL (the two most relevant classes of imitation learning are indicated in Fig. 15.1). Inverse RL addresses how to extract a suitable cost function from observed behavior, an idea going back to *Kalman* [15.8]. Algorithms for inverse RL are tightly coupled to RL which we can only briefly discuss in Sect. 15.3.6, *Curse of Goal Specification*, but they are also discussed in Sect. 75.4.3.

As learning has found many applications in robotics, this chapter can only scratch the surface and extensions of several sections can be found in [15.9, 10]. By focusing on the key approaches for teaching a robot new abilities, we nevertheless hope to give a concise overview.

15.1.2 Robot Learning Settings

The setting of robot learning methods is determined by the data. Data can either be obtained from an expert or generated by interaction with the system. Often data needs to be reproduced accurately (as e.g., for model learning or imitation learning by behavioral cloning) while in other cases a feedback signal can determine what data is actually relevant for the system (directly in RL and somewhat more indirectly in imitation learning by inverse RL). Also, the type of data generation will determine what robot learning methods are applicable as indicated in Fig. 15.1. For example, while expert data is crucial for imitation, it may simply be the wrong data to learn from in model learning.

Data for Robot Learning

In comparison to many other machine learning domains, robot learning suffers from a variety of complex real-world data problems. Real-world training time is limited – hence, only relatively few complete executions of a task can ever be generated. These episodes are frequently perceived noisily, have a fair amount of variability in the outcome, do not cover all possible scenarios and most often do not include all reactions to external stimuli. While episodes are scarce, the samples executed during a trial are not (but are often not helpful as they contain less information for some types of learning). Instead, high-dimensional data is obtained at a fast rate (e.g., proprioceptive information at 500 Hz to 5 kHz, vision at 30–200 Hz, tactile at 50 Hz to 5 kHz depending on the measurement mode). The data the robot can learn from is usually generated by the imperfect controller that needs to be improved. Thus, generating *interesting* data that has sufficiently useful information is a recurring problem in robot learning, discussed under various names, like exploration/exploitation tradeoff, persistent excitation, etc. Last not least, for high-dimensional robot systems (like humanoids), there is no *representative training data set* to learn from. A small change in posture of a robot can radically change in which part of the workspace the robot operates. Rapidly learning on the fly is frequently more important than squeezing the most of information from a few data points. Sharing data across different robots would be desirable, envisioning a *cloud robotics* approach. However, so far, even various instantiations of the same robot model can create quite different data depending on their wear and tear, calibration, and maintenance. Sharing data across different robot models has not seen a lot of success, as kinematic and dynamic differences can lead to rather different control and planning realizations. Thus, machine learning techniques for robotics

Avidbots Ex1016
Page 401 of 2154

become quickly rather specialized and domain/robotic specific.

What Feedback Is Available for Learning?

A straightforward way to categorize robot learning approaches is given by the type of feedback [15.11]. A scalar performance score such as a reward or cost will often result in a *RL* approach. A presented desired action or predicted behavior allow supervised learning approaches such as model learning or direct imitation learning. Feedback in terms of an explanation has become most prominent in inverse RL. These methods will be explained in more detail below. Note that unsupervised learning problems where no feedback is required also can be found in robotics [15.5, 12].

15.1.3 Notation

In the next section and in Fig. 15.1, we will use the following notation. In general, we follow the tradition in RL [15.13]. When it comes to specific control problems, we use $\boldsymbol{q}$ for a vector of states in configuration space (often called joint-space), $\boldsymbol{u}$ for motor commands in configuration space, and $\boldsymbol{x}$ for a state in task- or effector-space.

15.2 Model Learning

In recent years, methods to learn models from data have become increasingly interesting tools for robotics. There are a variety of reasons for this development. First, robot control – as needed in less structured dynamic and more stochastic environments – requires model-based control to realize high compliance, since stiff control without good models exposes the robot and its environment to more risk of damage. Second, due to the complexity of modern robot systems, standard models like rigid body dynamics are only rough approximations which do not accurately model unknown sources of nonlinearities from actuation (hydraulics and motor saturation), passive elements like wires, hydraulic hoses, cable stretch, and/or sources of friction and stiction. Finally, interaction with an unstructured, uncertain, and human-inhabited environment may require to learn models on the fly [15.14, 15]. In addition, wear and tear of a robot in continual use will create drifting models. All these issues can be addressed by learning models directly from measured data, ideally in a continual and incremental on-line fashion, using appropriate nonlinear function approximation techniques.

We discuss different types of models (forward models, inverse models, mixed models, and operator models) in Sect. 15.2.1, along with difference architectures for model learning in Sect. 15.2.2. Some of the challenges that arise in these models in the domain of robotics are discussed in Sect. 15.2.3. How models can be learned using machine learning techniques are described in Sect. 15.2.4, and examples where model learning has proven to be helpful for the action generation in complex robot systems [15.14, 16–21] are highlighted in Sect. 15.2.5.

While this chapter attempts to give a fairly exhaustive overview on model learning in robotics, it is of course not complete due to the space constraints of a book chapter. An extended version of this section can be found in [15.9].

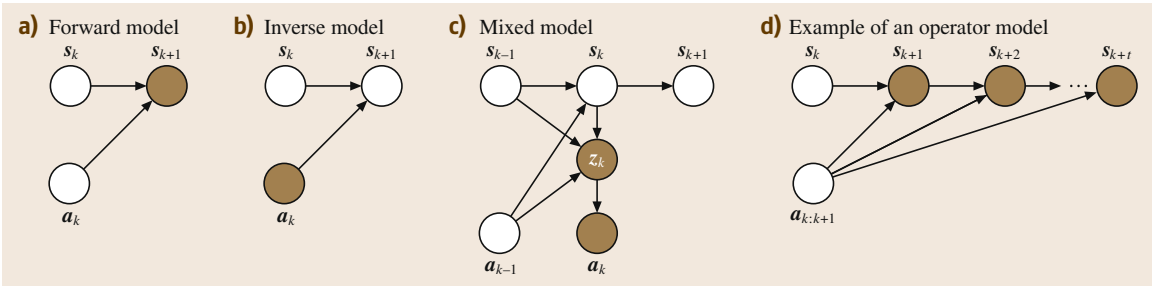


Fig.15.2a–d Graphical models for different types of models. The *white nodes* denote the *observed* quantities, while the *brown nodes* represent the quantities to be *inferred*. **(a)** The forward model allows inferring the next state given current state and action. **(b)** The inverse model determines the action required to move the system from the current state to the next state. **(c)** The mixed model approach combines forward and inverse models in problems where a unique inverse does not exist. Here, the forward and inverse models are linked by a latent variable z_k. **(d)** The operator model is needed when dealing with finite sequences of future states

15.2.1 Model Types

Models represent the functional relationship between input and output variables. If we can observe the current state $\boldsymbol{s}_k$ of a system and the action $\boldsymbol{a}_k$ currently applied to the system, we can attempt to predict the next state $\boldsymbol{s}_{k+1}$, which is called a forward model with mapping $(\boldsymbol{s}_k, \boldsymbol{a}_k) \rightarrow \boldsymbol{s}_{k+1}$. We can also use the inverse model to infer the current action, i. e., the relation $(\boldsymbol{s}_k, \boldsymbol{s}_{k+1}) \rightarrow \boldsymbol{a}_k$, if we know the current state and the desired or expected future state. There are also approaches combining forward and inverse models for prediction which we will refer to as mixed model approaches. However, for many applications the system behavior has to be predicted for the next t-steps rather than for the next single step; models that predict a series of states will be called operator models. Graphical depictions of these different models are shown in Fig. 15.2.

Forward Models

Forward models predict the next state of a dynamic system given the current action and current state. The forward model directly corresponds to the state transfer

$$\boldsymbol{s}_{k+1} = f_{\text{forward}}(\boldsymbol{s}_k, \boldsymbol{a}_k) \,. \tag{15.1}$$

This function expresses the causal physical properties of the system, and the learning of such causal mappings is a well-defined problem using standard regression techniques. While forward models are unique mappings in classical physics, there are several cases where forward models alone do not provide sufficient information to uniquely determine the next system's state [15.22]. For instance, when a pendulum is located at its unstable equilibrium point, it will tend to go either to the left or right rather than remain in the center. Nevertheless, the prediction of the forward model would be for the pendulum to remain centered. In this case, the state transfer function is not sufficient to capture the interesting dynamics of the system [15.22, 23]. For such cases, a probabilistic model $T(\boldsymbol{s}_{k+1}, \boldsymbol{a}_k, \boldsymbol{s}_k) = P(\boldsymbol{s}_{k+1}|\boldsymbol{s}_k, \boldsymbol{a}_k)$ may be more appropriate as it captures both likely modes with a conditional density instead of just the mean behavior.

An early application of forward models in classical control is the Smith predictor, where the forward model is employed to cancel out delays imposed by the feedback loop [15.24]. Subsequently, forward models have been applied in the context of model reference adaptive control (MRAC) [15.25]. MRAC is a control system in which the performance of an action is predicted using a forward reference model. The controller adjusts the action based on the resulting error between the desired and current state. Hence, the policy π for MRAC can be written as

$$\pi(\boldsymbol{s}) = \underset{\boldsymbol{a}}{\text{argmin}} \left(f_{\text{forward}}(\boldsymbol{s}, \boldsymbol{a}) - \boldsymbol{s}_{\text{des}}\right) , \tag{15.2}$$

where $\boldsymbol{s}_{\text{des}}$ denotes a desired trajectory and $\boldsymbol{s}$ represents the observed state. MRAC was originally developed for continuous-time system and extended later for discrete and stochastic systems [15.25]. Applications of MRAC can be widely found for robot control, such as adaptive manipulator control [15.26].

Further applications of forward models can be found in the wide class of model predictive control (MPC) [15.27]. MPC computes optimal actions by minimizing a given cost function over a certain prediction horizon N. The MPC control policy can be described by

$$\pi(\boldsymbol{s}) = \underset{\boldsymbol{a}_{t:t+N}}{\text{argmin}} \sum_{k=t}^{t+N} F_{\text{cost}} \left(f_{\text{forward}}(\boldsymbol{s}_k, \boldsymbol{a}_k), \boldsymbol{s}_{k,\text{des}}, \boldsymbol{a}_k\right) , \tag{15.3}$$

where $F_{\text{cost}}(\boldsymbol{s}, \boldsymbol{s}_{\text{des}})$ denotes the cost of an action $\boldsymbol{a}_k$ resulting in a state $\boldsymbol{s}$ away from the desired state $\boldsymbol{s}_{\text{des}}$. MPC was first developed for linear system models and, subsequently, extended to more complex nonlinear models with constraints [15.27]. Forward models have also been essential in model-based RL approaches, which relate to the problem of optimal control [15.28–30]. Here, the forward models describe the transition dynamics determining the probability of reaching the next state given current state and action. In contrast to previous applications, the forward models incorporate a probabilistic description of the system dynamics in this case [15.31, 32]. Example applications of forward models for optimal control are given in the case studies in Sect. 15.2.5, *Simulation-Based Optimization*.

Inverse Models

Inverse models predict the action required to move the system from the current state to a desired future state. In contrast to forward models, inverse models represent an anticausal system. Thus, inverse models may not exist or be well defined. However, for several cases, such as for inverse dynamics of nonoveractuated robots, the inverse relationship is well-defined. Ill-posed inverse modeling problems can be solved by introducing additional constraints, as discussed in Sect. 15.2.1, *Mixed Models*.

For control, applications of inverse models can be traditionally found in computed torque robot control, where the inverse dynamics model predicts the required torques, i. e., the action, to move the robot along a desired joint space trajectory. Thus, the computed torque

Avidbots Ex1016
Page 403 of 2154

control policy can be described by

$$\pi(\boldsymbol{s}) = f_{\text{inverse}}(\boldsymbol{s}, \boldsymbol{s}_{\text{des}}) + k(\boldsymbol{s} - \boldsymbol{s}_{\text{des}}) \,, \tag{15.4}$$

where the function $k(\boldsymbol{s} - \boldsymbol{s}_{\text{des}})$ is an error correction term (e.g., a proportional–derivative (PD)-controller since both positions and velocities are normally part of the state in a second order dynamical system) needed for stabilization of the robot. If an accurate inverse dynamics model is given, the predicted torques are sufficient to obtain a precise tracking performance. The inverse dynamics control approach is closely related to the computed torque control method.

Here, the feedback term acts through the inverse model of the system giving a control policy,

$$\pi(\boldsymbol{s}) = f_{\text{inverse}}\left[\boldsymbol{s}, \boldsymbol{s}_{\text{des}} + k(\boldsymbol{s} - \boldsymbol{s}_{\text{des}})\right] . \tag{15.5}$$

If the inverse model perfectly describes the inverse dynamics, inverse dynamics control will perfectly compensate for all nonlinearities occurring in the system. Control approaches based on inverse models are well-known in the robotics community, and have gained increasing popularity since the late 1980s, as the increase in computational power and the existence of torque-controlled robots have allowed computing more complex models for real-time control. The concept of feedback linearization is another, more general way to derive inverse dynamics control laws, offering possibly more applications for learned models [15.33, 34].

Mixed Models

In addition to forward and inverse models, there are also methods combining both types of models. Since modeling the forward relationship is well defined while modeling the inverse relation may be an ill-posed problem, the basic idea behind the combined mixed model is that the information encoded in the forward model can help to resolve the nonuniqueness of the inverse model. A typical ill-posed inverse modeling problem is the inverse kinematics of redundant robots. Given a joint configuration $\boldsymbol{q}$, the task space position $\boldsymbol{x} = f(\boldsymbol{q})$ can be determined exactly (i. e., the forward model is well defined), while there may be many possible joint configurations $\boldsymbol{q}$ for a given task space position $\boldsymbol{x}$, i. e., the inverse model could have infinitely many solutions and deriving an average suitable solution is not straightforward. The problem is that the set of inverse solutions may not form a convex set. Thus, when learning such inverse mappings, the learning algorithm will potentially average over the nonconvex set, giving rise to an invalid solution.

The ill-posedness of the inverse model can be resolved if it is combined with the prediction of a forward model, which checks whether the proposed inverse solution results in the desired state [15.35]. If not, the remaining error can be used to adjust the learned inverse model such that it chooses a valid solution.

The mixed model approach, i. e., the composite of forward and inverse models, was first proposed in conjunction with the distal teacher learning approach [15.35], and discussed in Sect. 15.2.2, *Distal Teacher Learning*. The proposed mixed models approach has subsequently evoked significant interest and has been extensively studied in the field of motor control [15.36, 37]. Furthermore, the mixed model approach is supported by some evidence that the human cerebellum can be modeled using forward–inverse composite models, such as MOSAIC [15.38, 39]. While mixed models have become well known in the neuroscience community, the application of such models in robot control is not yet widespread. Pioneering work on mixed models in the control community can be found in [15.40, 41], where mixed models are used for model reference control of an unknown Markov jump system. Despite that mixed model approaches are not widely used in control, with the appearance of increasingly more humanoid robots, biologically inspired robot controllers are gaining more popularity and the ones based on mixed models may present a promising approach [15.17, 42, 43].

Operator Models

The models from the preceding sections are mainly used to predict a single future state or action. However, in many problems, such as open-loop control, one would like to have information of the system for the next t-steps in the future. This problem is the multi-step ahead prediction problem, where the task is to predict a sequence of future values without the availability of output measurements in the horizon of interest. We will term the models which are employed to solve this problem as operator models. It turns out that such operator models are difficult to develop because of the lack of measurements in the prediction horizon. A straightforward idea would be to apply single-step prediction models t times in sequence, in order to obtain a series of future predictions. However, this approach is susceptible to error accumulation, i. e., errors made in the past are propagated into future predictions. An alternative is to apply autoregressive models which are extensively investigated in time series prediction [15.44], in order to employ past predicted values to predict future outcomes.

Combining operator models with control was first motivated by the need of extension of forward models for multi-step predictions [15.45]. In more recent work, variations of traditional ARX and ARMAX models for

Avidbots Ex1016
Page 404 of 2154

Table 15.1 Overview on model types associated with applicable learning architectures and example applications

Model type	Learning architecture	Example applications
Forward model	Direct modeling	Prediction, filtering, learning simulations, optimization
Inverse model	Direct modeling (if invertible), indirect modeling	Inverse dynamics control, computed torque control, feedback linearization control
Mixed model	Direct modeling (if invertible), indirect modeling, distal-teacher	Inverse kinematics, operational space control, multiple model control
Operator model	Direct modeling	Planning, optimization, model predictive control, delay compensation

nonlinear cases have been proposed for control [15.46, 47]. However, operator models based on some parametric structures, such as ARX or ARMAX, have shown to have difficulties in more complex systems. The situation is even worse in the presence of noise or complex nonlinear dynamics. These difficulties give reasons to employ nonparametric operator models for multi-step predictions [15.48, 49].

15.2.2 Model Learning Architectures

In the previous section, we have presented different prediction problems that require different types of models. A central question is how to learn and adapt these different models while they are being used. We will distinguish between direct modeling, indirect modeling, and the distal teacher approach. Table 15.1 shows an overview of model types associated with applicable learning architectures. These architectures can be observed in Fig. 15.3.

Direct Modeling

Direct learning is probably the most straightforward and most frequently applied way to obtain a model despite that it is not always applicable. In this learning paradigm, the model is directly learned by observing the inputs and outputs, creating a supervised learning problem. Direct model learning can be implemented using most regression techniques, such as linear regression, local linear regression, recursive regression [15.50, 51], neural networks [15.52, 53], or other statistical approximation techniques, currently mostly based on kernel methods [15.54, 55].

An early example for direct learning control was the self-tuning regulator that generates a forward model and adapts it online [15.56]. Assuming the estimated forward model is correct (also called the certainty equivalence principle), the self-tuning regulator will estimate an appropriate control law online. However, the forward model in the traditional self-tuning regulator has a fixed parametric structure and, hence, it cannot deal automatically with unknown nonlinearities [15.47, 57]. The main reason why parametric models need to be used in direct modeling techniques is that such model parameterization is necessary for a convenient formulation of the control law and, more importantly, for the rigorous stability analysis. As the parametric model structure is often too limited for complex robot systems, learning methods with more degrees of freedom

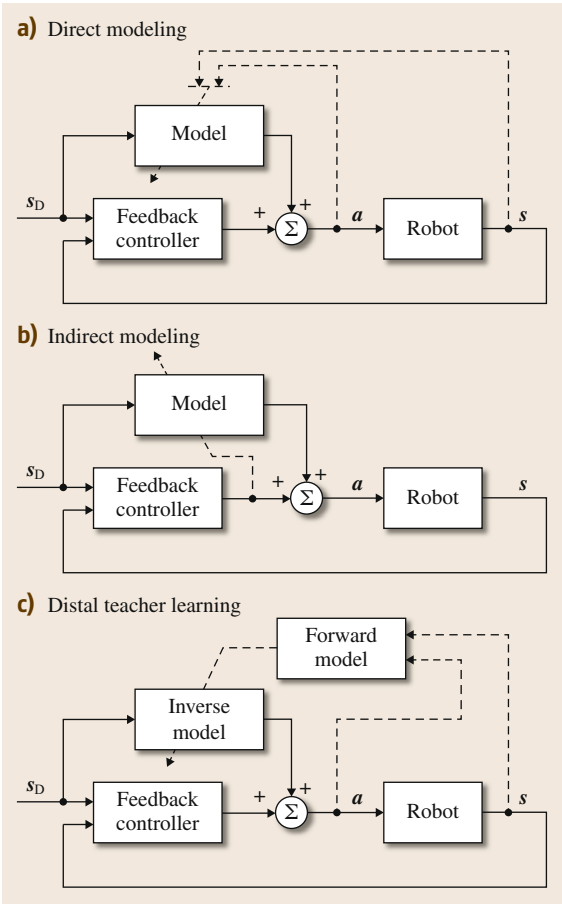


Fig.15.3a–c Learning architectures in model learning applied to control. **(a)** In the direct modeling approach, the model is learned directly from the observations – the given example shows learning and inverse model. **(b)** Indirect modeling approximates the model using the output of the feedback controller as error signal. **(c)** In the distal teacher learning approach, the inverse model's error is determined using the forward model. The resulting models will converge to an identity transformation

Avidbots Ex1016
Page 405 of 2154

are needed, such as neural networks or other machine learning techniques [15.58, 59]. However, including sophisticated learning algorithms in control makes the analysis more complex if not impossible. Some examples of nonparameteric learning for control have been applied to unmanned airplane control [15.60], where the learning system models can prevent the airplane from crashing in case of changes in the airplane dynamics, e.g., due to failure of the mechanics.

Inverse models can also be learned in a direct manner if the inverse mapping is well defined. A well-known example is the inverse dynamics model required by computed torque and inverse dynamics control [15.61, 62]. If direct modeling is applicable, learning becomes straightforward and can be achieved using standard regression techniques [15.14, 63, 64]. Early work in learning inverse models for control attempted to adapt a parametric form of the rigid body dynamics model which is linear in its parameters and, hence, can be estimated from data straightforwardly using linear regression [15.65–67]. In practice, the estimation of dynamics parameters is not always straightforward. It is hard to create sufficiently rich data sets so that physically plausible parameters can be identified [15.15], and when identified online, additional persistent excitation issues occur [15.68]. Models with fixed parametric structures are not capable of capturing certain nonlinearities; hence, more sophisticated models have been introduced for learning inverse dynamics, such as neural networks [15.64, 69] or statistical nonparametric models [15.14, 63, 70]. There have also been attempts to combine parametric rigid body dynamics model with nonparametric model learning for approximating the inverse dynamics [15.71]. Similar as in inverse dynamics models, feedback linearization can also be used in conjunction with direct model learning. Again, the nonlinear dynamics can now be approximated using neural networks or other nonparametric learning methods [15.60, 72]. Stability analysis of feedback linearization control with learned models is possible extending the cases where the nonlinear dynamics could not be canceled perfectly [15.60]. It should be noted that learning an inverse dynamics model $(\boldsymbol{s}, \boldsymbol{s}_{\text{des}}) \rightarrow \boldsymbol{a}$ can have very slow learning performance when the initially learned model is rather far from reality. In this case, the predicted action $\boldsymbol{a}$ will create outcomes that are very far away from $\boldsymbol{s}_{\text{des}}$, such that the learning system, when trying out the bad action $\boldsymbol{a}$, can hardly extract any useful information from this experience how to actually realize $\boldsymbol{s}_{\text{des}}$.

While direct learning is mostly associated with learning a single type of model, it can also be applied to mixed models and operator models. The mixed model approach (e.g., combining inverse and forward models) finds its application in learning control for multiple-module systems. The basic idea is to decompose a complex system into many simpler subsystems which can be controlled individually [15.41]. The problem is how to choose an appropriate architecture for the multiple controllers, and how to switch between the multiple modules. Following the multiple-module idea, each controller module consists of a pair of inverse and forward models in the mixed model approach. The controller can be considered as an inverse model, while the forward model is essentially used to switch between the different modules [15.36]. Such multiple pairs of forward and inverse models can be learned directly using gradient-descend methods or expectation-maximization [15.42]. Direct learning of operator models has been done with neural networks [15.73]. In more recent work, probabilistic methods are employed to learn such operator models [15.48, 49].

Indirect Modeling

Direct learning works well when the functional relationship is well defined. However, there can be situations where the inverse relationship is not well defined, such as in the differential inverse kinematics problem. In such cases, inverse models can often still be learned, but in an indirect way. One such indirect modeling technique is feedback error model learning [15.74]. Feedback error learning relies on the output of a feedback controller that is used to generate the error signals employed to learn the feedforward controller. In problems, such as feedforward inverse dynamics control [15.61], this approach can be understood particularly well: if the inverse dynamics model is perfect, the feedback controller is not needed and its output will be zero. If it is nonzero, it corresponds to the error of the feedforward model [15.61]. The intuition behind feedback error learning is that by minimizing the feedback errors for learning the inverse model, the feedback control term will be decreasing as the model converges.

Compared to the direct model learning, feedback error learning is a goal-directed model learning approach resulting from the minimization of feedback error – in this error can only be zero if the goal was achieved. Another important difference between feedback error learning and direct learning is that feedback error learning has to perform online, while direct model learning can be done both online and offline. Additionally, feedback error learning does not create supervised learning data in the classical sense: the feedback error is not computed from an output target, but just from an error that represents the gradient toward better performance. This issue, and additionally a complex interaction between PD gains and learning stability [15.75] can make

Avidbots Ex1016
Page 406 of 2154

it more complex to set up a successful feedback error learning system.

Feedback error learning is biologically motivated due to its inspiration from cerebellar motor control [15.37]. It has been further developed for control with robotics applications, originally employing neural networks [15.76, 77]. Feedback error learning can also be used with various nonparametric learning methods [15.75]. Conditions for the stability of feedback error learning control in combination with nonparametric approaches have also been investigated [15.75].

Indirect model learning can also be used in the mixed model approach [15.78]. Here, the attempt has been made to combine the feedback error learning with the mixture of experts' architecture to learn multiple inverse models for different manipulated objects, where the inverse models are learned indirectly using the feedback error learning approach [15.78]. The forward model is used for training a gating network, as it is well defined. The gating network subsequently generates a weighted prediction of the multiple inverse models, where the predictors determine the locally responsible models.

Distal Teacher Learning

The distal teacher learning approach was motivated by the necessity to learn general inverse models which suffer from the ill-posedness problem [15.35]. Here, the nonuniqueness of the inverse model is resolved by the distal teacher which determines the error of the inverse model using a forward model. Unlike the feedback error learning approach, this insight allows directly aiming at a globally consistent inverse model instead of local on-policy optimization. The distal teacher employs two interacting learning process: one process where the forward model is learned, and the other process where the learned forward model is used for determining the error of the inverse model. In the original distal learning approach, the inverse model's output is validated by the forward model, as the composite of these models yields an identity mapping if perfectly learned [15.35]. The intuition behind this approach is that the inverse model will learn a correct solution by minimizing the error between the output of the forward model and the input to the inverse model. Thus, the inverse model will result in solutions that are consistent with the forward model. The distal teacher approach has successfully learned particular solutions for multi-valued mappings, such as inverse kinematics of redundant robots [15.35]. Similar to feedback error model learning [15.74], distal teacher learning is also a goal-directed learning method applicable for various robot control scenarios.

The distal learning approach is particularly suitable for control when combining with the mixed models, as it naturally incorporates the mixed model principle. The distal teacher learning approach with mixed models has motivated a number of follow-up projects with several robot control applications [15.17, 43, 79].

15.2.3 Model Learning Challenges and Constraints

Given different types of models and their associated learning architectures, it may not be straightforward to apply machine learning methods [15.54, 55, 80] to these models in robotics. Several important problems arise that need to be addressed before learning methods can be successfully applied, and are summarized in Table 15.2.

Data Challenge

In order to learn a *good* model for applications in robotics, the sampled data has to cover a preferably large region of the model state space, and ideally cover it densely. Of course, in high-dimensional systems, one can never cover the complete state space – indeed, in complex robots like humanoids with state spaces of over 30 dimensions, only a very tiny fraction of the possible space can be explored in a lifetime. Thus, one can only hope to sample data in a task specific way, which is usually just in a very small part of the entire data space. For the purpose of generalization and accuracy, the data has to be sufficiently rich, i. e., it should contain as much information about the system as possible. Thus, generating large and rich data sets for model learning is an essential step. This step often requires additional excitation of the robot system during data generation, analogous to persistent excitation in classical system identification [15.68].

Table 15.2 Some challenges for model learning in robotics

Data challenges	Algorithmic constraints	Real-world challenges
High-dimensionality	Incremental updates	Safety
Smoothness	Real-time	Robustness
Richness of data	Online learning	Generalization
Noise	Efficiency	Interaction
Outliers	Large data sets	Stability
Redundant data	Prior knowledge	Uncertainty
Missing data	Sparse data	In the environment

Avidbots Ex1016
Page 407 of 2154

Many machine learning methods for regression assume that the model to be learned is sufficiently smooth. However, there are many applications in robotics where the approximated functions are known to be nonsmooth – one example is a stiction-friction model. Such nonsmooth functions are sometimes approximated using kernel methods, where special kernels need to be defined in order to take the expected nonsmoothness in account [15.54, 55]. Discontinuities in a function can also be approximated by learning how to switch discontinuously between local models [15.81].

Difficulties arising from the large number of degrees of freedom in robotics can also be mitigated using dimensionality reduction methods [15.82, 83]. Dimensionality reduction is based on the insight that the useful information in the data may be represented by a low-dimensional manifold of the original input space. Dimensionality reduction methods have proven to be a powerful method for model learning in high-dimensional robot systems [15.63, 84].

As the data is sampled over a possibly long period of time in many robotics applications [15.85], problems of redundant and irrelevant data can occur. Redundant and irrelevant data can bias the model which severely degenerates the generalization performance. To overcome these issues, data needs to be filtered that is informative for the learning process [15.70, 86]. Dealing with noise and outliers is always a challenging problem for robot learning. Recent work in the machine learning community has addressed this problem by developing regularization frameworks based on statistical learning theory. The basic idea is to constrain the model to be learned in order to alleviate the contributions made by the noisy components in the data. Examples of regularized model learning methods include support vector regression and Gaussian process regression [15.87, 88]. Methods that explicitly account for noise in the data can significantly improve their generalization performance.

Algorithmic Constraints

In many scenarios, robot learning algorithms have to deal with massive amounts of data coming from their sensors. Thus, algorithms need to be efficient in terms of computation without sacrificing learning accuracy [15.89]. Fast and real-time computation presents a challenging problem. Standard model learning approaches, such as Gaussian process regression, for example, scale cubically in the number of training data points, preventing a straightforward usage in robotics. Sparse and reduced set methods smartly reduce the size of training data and, thus, decrease the computational effort for the learning and the prediction step [15.90]. Efficient implementations using parallel computation may also be used to speed up machine learning algorithms [15.91].

Online learning is also needed for continuous adaptation to a changing world which is essential to make robots more autonomous [15.85]. Most machine learning methods are developed for learning in batch mode, while online learning requires incremental approximation of the model [15.14, 92]. An advantage of online model learning is the ability to uncover interesting state-space regions during task execution.

For some robots, it may be difficult or expensive to obtain a lot of data for learning. In these cases, we need algorithms which allow us to incorporate additional prior knowledge to improve the learning performance in the presence of sparse data [15.93, 94]. In kernel methods, prior knowledge can be specified by feature vectors which can be used to define appropriate kernels [15.93]. In contrast, probabilistic frameworks allow specifying a priori information [15.54]. Furthermore, prior knowledge, if given as a parametric model, can be inserted into nonparametric models yielding semiparametric learning approaches [15.71, 95]. Semiparametric models have shown to be capable in learning competitive models, when little data is available [15.71].

Real-World Constraints

Ensuring safe interaction of robots with human beings in everyday life, machine learning algorithms developed for robotics applications have to minimize the risk of injury and damage. For critical applications, such as medical or service robotics, robustness and reliability are among the most important criteria which have to be fulfilled by model learning. Learning becomes more robust by employing feature selection as a preprocessing step, and has been used in several robotics application in vision and control [15.70, 96].

Robustness also requires the ability of the learning algorithm to deal with missing data. Missing data due to measurement errors poses a difficult problem for model learning in robotics. Recently, probabilistic learning methods have been used to infer the missing components in the training data [15.97]. An advantage of probabilistic methods comes from the ability to assign uncertainty to the predicted values, making it easier to deal with missing and imprecise data.

15.2.4 Model Learning Methods

We consider model learning in the context of supervised learning, where the inputs $\boldsymbol{x}$ and outputs $\boldsymbol{y}$ are given. The true output data is corrupted by noise $\boldsymbol{\epsilon}$, i. e.,

$$\boldsymbol{y} = f(\boldsymbol{x}) + \boldsymbol{\epsilon} \,. \tag{15.6}$$

Avidbots Ex1016
Page 408 of 2154

Table 15.3 A large variety of machine learning methods have been applied in model learning for robotics. We distinguish between global and local methods, as well as semilocal methods which combine both approaches. The methods differ in the training mode and their online capabilities. For computational complexity, n denotes the total number of training points, m is the number of data points in a local model, and M is the number to local models

Method	Type	Mode	Online	Complexity	Learning applications
Locally weighted projection regression [15.80]	Local	Incremental	Yes	$\mathcal{O}(n)$	Inverse dynamics [15.63], Foothold quality model [15.21]
Local Gaussian process regression [15.14]	Local	Incremental	Yes	$\mathcal{O}(m^2)$	Inverse dynamics [15.14]
Gaussian mixture model [15.98]	Semilocal	Batch	No	$\mathcal{O}(Mn)$	Human motion model [15.99]
Bayesian committee machine [15.100]	Semilocal	Batch	No	$\mathcal{O}(m^2n)$	Inverse dynamics [15.54]
Sparse Gaussian process regression [15.101]	Global	Incremental	Yes	$\mathcal{O}(n^2)$	Transition dynamics [15.32], Task model [15.102]
Gaussian process regression [15.103]	Global	Batch	No	$\mathcal{O}(n^3)$	Terrain model [15.104], State estimation model [15.105]
Support vector regression [15.55]	Global	Batch	No	$\mathcal{O}(n^2)$	ZMP control model [15.106], Grasp stability model [15.107]
Incremental support vector machine [15.108]	Global	Incremental	Yes	$\mathcal{O}(n^2)$	Inverse dynamics [15.109]

Note that the noise is additive to $f(\boldsymbol{x})$; models that address noise in the inputs can be developed as well [15.110]. The performance of a model is given by its *generalization* ability, i. e., the learned model should be able to provide precise predictions of the output on new input data. Different learning techniques make different assumptions on how to model the function f given the data. From a function approximation point of view, we will distinguish between *global* and *local* techniques. Beyond the local and global types of model learning, there are also approaches that combine both ideas. An example of such a hybrid approach is the mixtures of experts [15.98, 111]. Here, the data is partitioned into smaller local models by a learned gating network, and prediction is accomplished by fusing local model predictions. Mixture of experts have been further embedded into a Bayesian learning framework, giving rise to a number of Bayesian hybrid approaches such as committee machines [15.100], mixtures of Gaussian models [15.112], or infinite mixtures of experts [15.113].

Global Regression

Global regression techniques model the underlying function f using *all* observed data [15.114], i. e., every data point is potentially adjusting every parameter of the learning system. A straightforward way to model the function f is to assume a parametric structure, such as linear or polynomial models or multilayer perceptron neural networks, and to fit the model parameters using training data [15.52, 114, 115]. However, fixing the model with a parametric structure beforehand may not suffice to explain the sampled data with sufficient accuracy, which motivates nonparametric model learning frameworks [15.54, 55, 115]. Nonparametric learning does not mean that there are no parameters, just that the correct number of relevant parameters is unknown, and that it is unlikely that a finite set of parameters would suffice to model the entire regression problem.

In the last two decades, parametric and nonparametric regression methods have been focussing on functions f modeled as

$$f(\boldsymbol{x}) = \boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}) \,, \tag{15.7}$$

where $\boldsymbol{\theta}$ is a weight vector and $\boldsymbol{\phi}$ is a vector of nonlinear functions, also called *features* or *basis-functions*. $\boldsymbol{\phi}$ projects $\boldsymbol{x}$ into some high-dimensional space, in which the learning problem is assumed to become linear in the parameters. The basic idea behind nonparametric regression is that the optimal model structure should be obtained from the training data. Hence, the size of the weight vector $\boldsymbol{\theta}$ is not fixed but can increase with the number of training data points. In contrast to nonparametric statistical approaches, other popular function approximation methods, such as neural networks, fix the model structure beforehand [15.52], like the number of nodes and their connections. However, there is a connection between artificial neural networks and probabilistic approaches [15.116, 117] using the formalization in (15.7).

Learning systems as formalized in (15.7) are mostly discussed as kernel methods [15.55]. Learning a model is equivalent to finding a tradeoff between the model's complexity and the best fit of the model to the observed data. Thus, it is desirable to have a model which is simple but at the same time can explain the data well. The number of features (i. e., the length of feature vector $\boldsymbol{\phi}$) has a direct correlation with the complexity of the regression model, as the more features one allows, usually the more powerful the learning system

Avidbots Ex1016
Page 409 of 2154

becomes, but it also takes more data to constrain the open parameters of the model. One of the key insights of kernel methods [15.54, 55] was that one can actually work with *infinitely* many features using the so-called *kernel trick*, where the formulation of the regression problem only involves inner products of the feature vectors $\boldsymbol{\phi}(\boldsymbol{x}_i)^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_j)$ of any two inputs $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, and that this inner product can often be computed in closed form by a kernel function $k(\boldsymbol{x}_i, \boldsymbol{x}_j)$. This approach introduces a dual form of the regression learning problem where the number of open parameters is identical to the number of n training data, and not the number of features. In support vector regression (SVR) [15.55], a sparsification process will choose the minimal number of parameters needed to represent the training data. In Gaussian process regression [15.54], a probabilistic approach is taken that places a prior distribution over $\boldsymbol{\theta}$. The posterior parameters can subsequently be obtained by optimizing the corresponding marginal likelihood.

Kernel and probabilistic methods have proven to be successful tools for model learning over the last decade, resulting in a number of widely applied regression methods [15.87, 88, 118]. These methods can be applied to high-dimensional data. They can also deal well with noisy data, as the noise is taken in account indirectly by regularizing the model complexity. Furthermore, they are relatively easy to use, as several black-box implementations are available. However, the major drawback of these methods are the computational complexity. For instance, for Gaussian process regression the complexity scales cubically in terms of the number of training data points. Thus, one active research line in machine learning is to reduce the computational cost of those approaches. Due to customized techniques for robotics, kernel and probabilistic regression techniques found their ways into several robotics applications, such as robot control [15.70], sensor modeling [15.119], or state estimation [15.105].

Local Learning Methods

The basic idea of local regression techniques is to estimate the underlying function f *within a local neighborhood* around a query input point $\boldsymbol{x}_{\mathrm{q}}$. The data points in this neighborhood can then be used to predict the outcome for the query point. Generally, local regression models can be obtained by minimizing the following cost function J using n training data points

$$J = \sum_{k=1}^{n} w\left[(\boldsymbol{x}_k - \boldsymbol{x}_{\mathrm{q}})^{\mathrm{T}}\boldsymbol{D}(\boldsymbol{x}_k - \boldsymbol{x}_{\mathrm{q}})\right] \|\boldsymbol{y}_k - \hat{\boldsymbol{f}}(\boldsymbol{x}_k)\|^2 \,. \tag{15.8}$$

As indicated by (15.8), the essential ingredients for a local regression model are the neighborhood function w and the local model $\hat{f}$. The neighborhood function w, which is controlled by a distance metric $\boldsymbol{D}$, often simplified as a scalar width parameter $\boldsymbol{D} = 1/h\boldsymbol{I}$, measures the distance between a query point $\boldsymbol{x}_{\mathrm{q}}$ to the points in the training data. The local model $\hat{f}$ describes the function structure used to approximate f within the neighborhood around $\boldsymbol{x}_{\mathrm{q}}$ [15.120, 121]. Depending on the complexity of the data, different function structures can be assumed for the local model $\hat{f}$, such as a linear or a polynomial model. The open-parameters of $\hat{f}$ can be estimated straightforwardly by minimizing J with respect to these parameters.

The choice of the neighborhood function and its distance metric parameter is more involved. Several techniques have been suggested for estimating $\boldsymbol{D}$ for a given w, including the minimization of the leave-one-out cross-validation error and adaptive bandwidth selection [15.122, 123].

Due to their simplicity and computational efficiency, local regression techniques have become widespread in model learning for robotics [15.124–126]. In the last decade, novel local regression approaches were further developed in order to cope with the demands in real-time robotics applications, such as locally weighted projection regression [15.92, 127]. Inspired by local regression techniques, these methods first employ a partitioning of the input space into smaller local regions, for which locally linear models are approximated. A useful metaphor is to imagine that the input space is partitioned into disjoint boxes, and for the data points falling into each of these boxes, an independent local model is fit, usually of very simple complexity, like locally linear models.

In addition to the computational efficiency, local methods can deal with less smooth functions and do not require the same smoothness and regularity conditions as global regression methods. However, it has been shown in practice that local methods suffer from problems induced by high-dimensional data, as notions of locality break down for sparse, high-dimensional data. Furthermore, the learning performance of local methods may be sensitive to noise and heavily depends on the way how the input space is partitioned, i. e., the configuration of the neighborhood function w. These problems still present an active research topic [15.17, 128].

Several attempts have been made to scale local regression models to higher dimensional problems as required for many modern robotics systems. For example, locally weighted projection regression combines local regression with dimensionality reduction by projecting the input data into a lower dimensional space, where local regression is employed afterward [15.92]. Other methods combine nonparametric

Avidbots Ex1016
Page 410 of 2154

probabilistic regression, such as Gaussian process regression, with the local approaches while exploiting the strength of probabilistic methods for model learning in high-dimension [15.14, 17]. Local learning methods are very well suited to parallelization on modern computer hardware, as each local model can be fit independently.

15.2.5 Applications of Model Learning

Several representative examples of model learning are illustrated in the following sections.

Simulation-Based Optimization

As forward models directly describe the causal dynamic behavior of a system, learning such models has received much attention in model-based control, particularly for optimal control problems. In optimal control, a forward model is needed to predict the behavior of the control system over an extended time horizon in order to optimize the control policy. If an analytical model is inaccurate, learning the model offers an interesting alternative.

Atkeson and *Morimoto* [15.29] have been among the first to explore this approach using differential dynamic programming [15.129] for optimizing open-loop control policies. The basic idea was to use receptive field-weighted regression (a type of locally weighted regression) to learn the models of the cost function and the state transition. Differential dynamic programming locally linearizes the state transition model and generates a local quadratic approximation of the cost. These approximations are used to improve an open-loop policy where the linearizations are also updated after every policy update [15.29]. Atkeson and Morimoto used the method to learn the underactuated pendulum swing up task, where a pole is attached to with a passive joint to the endeffector of a robot arm. The goal of the robot is to bring the pole from a hanging to an upright position. Hence, the system needs to *pump* energy into the pendulum in order to swing it up (Fig. 15.4a). Subsequently, the pole needs to be stabilized at the upright position [15.130]. Starting from an unconstrained human demonstration, the robot was able to successfully learn the swing up and balance task after three trials [15.130]. The local trajectory optimization technique has been further extended to biped robot walking [15.131]. More recently, a related approach with parametric function approximation has been applied by *Abbeel* et al. [15.132] to learn autonomous helicopter flight (VIDEO 353).

While *Atkeson* and *Morimoto* [15.29] and *Abbeel* et al. [15.132] used the forward model as an implicit simulator, *Ng* et al. [15.30] used it as an explicit simulator (as originally suggested by *Sutton* [15.28] in form of the DYNA model). Here, with the help of the forward model, complete trajectories (or roll-outs) were simulated. The predictions of the forward model were perturbed by Gaussian noise with a repeating, fixed noise history. This injection of noise requires the learned controller to be robust to noise and modeling errors, while the re-use of the noise history reduces the variance of policy updates and results in much faster learning [15.133, 134]. In this way, roll-outs from a similar set of start-states are generated. The resulting performance of different control policy can be compared, which allows policy updates both by pairwise comparison or by gradient-based optimization. The approach has been able to successfully stabilize an helicopter in an inverted flight (Fig. 15.4b and VIDEO 352). A few similar examples in model predictive control (Sect. 15.2.1) exist, which employ a variety of different learning approaches such as statistical learning methods [15.48], neural networks [15.135], both for robot navigation [15.136] and helicopter control [15.137].

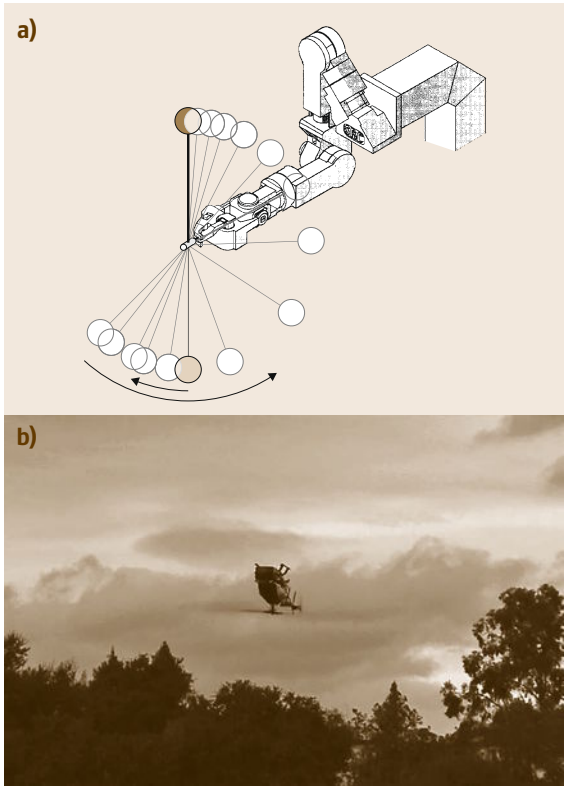


Fig. 15.4 (a) Sketch of learning a pendulum swing up task, and (b) inverse helicopter flight (courtesy of Andrew Y. Ng). In both the cases, the forward model is used to learn the dynamics of the system for policy optimization

Approximation-Based Inverse Dynamics Control

Inverse models, such as inverse dynamics models, are frequently used in robotics for feedforward control [15.62]. Inverse dynamics models characterize the required joint torques $\boldsymbol{\tau} = f(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}})$ to achieve a desired robot acceleration $\ddot{\boldsymbol{q}}$ for the state $\boldsymbol{q}, \dot{\boldsymbol{q}}$, i. e., the vectors of joint positions and velocities. In classical robot control, analytical dynamics models are derived in the framework of rigid body dynamics

$$\boldsymbol{\tau}\,(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}) = \mathbf{M}\,(\boldsymbol{q})\,\ddot{\boldsymbol{q}} + \boldsymbol{F}\,(\boldsymbol{q}, \dot{\boldsymbol{q}})\,, \tag{15.9}$$

where $\mathbf{M}(\boldsymbol{q})$ is the generalized inertia matrix of the robot, $\boldsymbol{F}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ is a vector defined by forces/torques from Coriolis effects, centripetal effects, and gravity. This model relies on many assumptions, such as rigidity of the robot's links, negligible nonlinearities from other sources (e.g., actuators), second-order dynamics, etc. [15.62]. However, modern robotics systems with highly nonlinear components, such as hydraulic tubes or elastic cable drives, can no longer be accurately modeled with rigid body dynamics. An inaccurate dynamics model can lead to severe losses in control performance and, in the worst case, instability. Instead of modeling the inverse dynamics manually based on physics and human insight, an inverse dynamics model can be learned from sampled data. Such a data-driven approach has the advantage that all nonlinearities encoded in the data will be approximated by the model [15.14].



Fig. 15.5 The 7 degree of freedom anthropomorphic SARCOS arm used in learning inverse dynamics, inverse kinematics, and operational space control

In most robots, the inverse dynamics model is a unique mapping from joint space into torque space, and learning inverse dynamics models is a standard regression problem. In order to generalize the learned models for a larger state space and to adapt the models for time-dependent changes in the dynamics, real-time online learning becomes necessary. However, online learning poses difficult challenges for any regression method. These problems have been addressed by real-time learning methods such as locally weighted projection regression [15.80]. *Nguyen-Tuong* and *Peters* [15.14, 70] combined the basic ideas behind the locally weighted projection regression method with Gaussian process regression [15.54], attempting to get as close as possible to the speed of local learning while having a comparable accuracy to Gaussian process regression. The resulting method has shown to be capable of real-time online learning of the robot's inverse dynamics. Instead of using local models, data sparsification methods can be employed to speed up kernel regression approaches for real-time scenarios [15.70].

It is worth noting that inverse dynamics model learning approaches can also be motivated from a biological point of view. *Kawato* [15.37] suggested that the cerebellum may act as an inverse dynamics model. Motivated by this insight, *Shibata* and *Schaal* [15.76] proposed a biologically inspired vestibulo-oculomotor control approach based on feedback-error learning of the inverse dynamics model. The problem is to stabilize the gaze of an oculomotor system that is mounted on a moving body, which creates permanent perturbations to eye (or camera) system. The cerebellum is known to predict the forces required to keep the image stabilized on the retina, based on efferent motor signals and inputs from the vestibular system. *Vijayakumar* and *Schaal* employ the locally weighted projection regression approach to learn the inverse model of the eye dynamics online [15.80]. The same locally weighted projection regression technique has also been used to learn a complete inverse dynamics model for a humanoid robot [15.63] (Fig. 15.5).

Learning Operational Space Control

Operational space control (OSC) allows a robot to accomplish a task space controller, e.g., for following a desired trajectory in task space or for impedance control [15.15, 138]. The relationship between the task space and joint space of the robot is well defined by the classical forward kinematics models $\boldsymbol{x} = f(\boldsymbol{q})$, where $\boldsymbol{q}$ denotes a joint space configuration and $\boldsymbol{x}$ represents the corresponding task space position. The task space velocity and acceleration are given by $\dot{\boldsymbol{x}} = \mathbf{J}(\boldsymbol{q})\dot{\boldsymbol{q}}$ and $\ddot{\boldsymbol{x}} =$

Avidbots Ex1016
Page 412 of 2154

$\dot{\mathbf{J}}(\boldsymbol{q})\dot{\boldsymbol{q}} + \mathbf{J}(\boldsymbol{q})\ddot{\boldsymbol{q}}$, respectively, where $\mathbf{J}(\boldsymbol{q}) = \partial f / \partial \boldsymbol{q}$ is the Jacobian. To obtain the joint torques required for task space control, the dynamics model (as given in (15.9)) is needed. The direct combination of dynamics and kinematics model yields one possible operational space control law

$$\boldsymbol{a} = \mathbf{M}\mathbf{J}_{\mathrm{W}}^{+}(\ddot{\boldsymbol{x}} - \dot{\mathbf{J}}\dot{\boldsymbol{q}}) + \boldsymbol{F}\,, \tag{15.10}$$

where $\mathbf{J}_{\mathrm{W}}^{+}$ denotes the weighted pseudoinverse of $\mathbf{J}$ [15.139] and $\boldsymbol{a}$ represents the desired joint torques. Equation (15.10) can be employed to generate the joint torques necessary for tracking a task space trajectory determined by a reference task-space acceleration [15.15]. Note that in practical applications, such control laws often require further terms, such as a null-space control law for joint-space stabilization [15.15].

As discussed before, dynamics models can be hard to obtain and, thus, learning can be a promising alternative. Learning an operational space control law corresponds to learning an inverse model such as $(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{x}}) \rightarrow \boldsymbol{a}$ [15.43]. However, learning such OSC models is an ill-posed problem, as there are infinitely many possible inverse models, due to the redundancy of joint space to task space mappings. For example, infinitely many solutions for a redundant robot can be derived analytically by varying the metric W of the weighted pseudoinverse in (15.10). As the space of possible solutions is not convex (Sect. 15.2.1, *Mixed Models*), such OSC models cannot be straightforwardly learned using regression models (unless the system has no redundant degrees of freedom). Similar problems appear in the limited case of differential inverse kinematics.

Both *D'Souza* et al. [15.79] and *Peters* and *Schaal* [15.43] noticed that local linearizations of the mapping in (15.10) will always form a locally convex space. Hence, data sets generated by such systems will also be locally convex. They furthermore realized that the predictive abilities of forward models allows determining local regions, where a locally consistent inverse model can be learned. However, depending on how the learning data was collected, i. e., the local data distribution, learning may result in rather different solutions to the local inverse model. Thus, two neighboring local inverse models may have quite different policies how to resolve redundancy, resulting in a discontinuous jump when transitioning from one local model to the next during control. As a result, a globally consistent solution to OSC can no longer be ensured. This insight leads to two significantly different approaches. *D'Souza* et al. [15.79] created a differential inverse kinematics learning system (i. e., a limited special case of an operational space control law) and chose to bias the learning system by selectively generating data. However, such an approach will generically be limited by the tradeoff between this intentional bias and the inverse model's accuracy. *Peters* and *Schaal* [15.43] treated learning of complete operational space control laws. They realized that a re-weighting of the data using an additional reward function allows regularizing these inverse models toward a globally consistent solution. Inspired by a result in analytical OSC [15.139], they suggest appropriate reward functions both for learning full OSC and differential inverse kinematics. The resulting mapping was shown to work on several robot systems. *Ting* et al. [15.17] presented an implementation of *Peters* and *Schaal* [15.43] approach with modern Bayesian machine learning, which sped up the performance significantly.

Instead of learning a direct OSC control law as done by *Peters* and *Schaal* [15.43], *Salaun* et al. [15.140] attempted to learn the well-defined differential forward kinematics as a first step (i. e., learning the Jacobian) using locally weighted projection regression. The corresponding weighted pseudoinverse of the Jacobian is subsequently computed using singular value decomposition (SVD). The obtained differential inverse kinematics model is combined with an inverse dynamics model to generate the joint space control torques [15.140]. Approximating inverse kinematics models has also been investigated in neural network learning and reservoir computing architectures [15.35, 141].

15.2.6 Future Directions

A key question in robotics is how to deal with uncertainty in the environment. As probabilistic machine learning techniques (e.g., Bayesian inference) have reached a high level of maturity [15.54], it has become clear how beneficial Bayesian machine learning can be for robot learning, especially, for model learning in the presence of uncertainty. However, machine learning techniques based on Bayesian inference are known to suffer from high computational complexity. Thus, special approximations will be needed as illustrated by the problems discussed in Sect. 15.2.3. Recently developed approximate inference methods such as in [15.90, 142] may become interesting new tools for robotics.

In order for robots to enter everyday life, they need to continuously learn and adapt to new tasks. Recent research on learning robot control has predominantly focused on learning single tasks that were studied in isolation. However, there is an opportunity to transfer knowledge between tasks, which is known as transfer learning in the field of machine learning [15.143]. To achieve this goal, robots need to learn the invariants of the individual tasks and environments and, sub-

sequently, exploit them when learning new tasks. In this context, similarities between tasks also need to be investigated and how they can be exploited for generalization [15.144].

In most of the model learning approaches, supervised learning methods are used. However, for many robot applications, target outputs are not always available. In these scenarios, semisupervised learning techniques can be useful to learn such models [15.145]. Semisupervised learning employs labeled as well as unlabeled data for model learning and can help to overcome the sparse labeling problem. It would also be beneficial to develop online versions of semisupervised approaches for real-time adaptation and learning.

Approximation-based control often still suffers from a lack of proper analysis of stability and convergence properties. Modern statistical learning theory might offer new tools to derive appropriate error bounds [15.55]. For example, generalization bounds can be used to estimate the learning performance of the controller and, from this insight, further statements and conditions about stability could be made.

Learning nonunique and ill-posed mappings is a key problem for several robot applications. This problem can be considered as a nonunique labeling problem, and statistical machine learning techniques, such as conditional random fields [15.146], may help to solve the problem. A conditional model specifies the probabilities of possible target outputs given an input observation sequence. As the target outputs are conditioned on the current input observations, nonuniqueness in the mappings can be resolved. It would be beneficial to investigate how such models could be incorporated into learning control and hierarchical control frameworks.

15.3 Reinforcement Learning

Probably the most general frameworks for learning control is RL which enables a robot to autonomously discover optimal behaviors through trial and error interactions with its environment. No explicit teaching signal is provided, as in supervised learning, but rather learning is based on a scalar objective function that measures the overall performance of the robot.

Consider the example of training a robot to return a table tennis ball ([15.147], VIDEO 354). In this case, the robot makes observations of dynamic variables specifying ball position and velocity as well as its internal dynamics including joint positions and velocities. These measurements capture the relevant *state* $\boldsymbol{s}$ of the system. The *actions* $\boldsymbol{a}$ available to the robot are the torque commands sent to the motors. A function π that computes the motor commands (i. e., the actions) based on the current state is the *policy*. The RL problem is to find a policy that optimizes the long term sum of *rewards* $R(\boldsymbol{s}, \boldsymbol{a})$, as already indicated in Sect. 15.1.3. The reward function in this example could be based on the number of successful hits, or on secondary criteria like energy consumption; RL algorithms aim to find (near-)optimal policies that maximize the reward function over a chosen time horizon. An extended discussion of robot RL techniques, on which this chapter is based, can be found in [15.10].

15.3.1 Relation to Classical Control

RL is closely related to the theory of classical optimal control, as well as dynamic programming, stochastic programming, simulation optimization, stochastic search, and optimal stopping [15.152]. Both RL and optimal control address the problem of finding an optimal policy (often also called the *controller* or *control policy*) that optimizes an objective function over a time horizon, and both rely on the notion of a system being described by an underlying set of states, controls and a plant or model that describes transitions between states. However, optimal control assumes perfect knowledge of the system's description in the form of a model (e.g., a model as in Sect. 15.2 that describes what the next state of the robot will be given the current state and action). For such models, optimal control ensures strong guarantees which, nevertheless, often break down due to model and computational approximations.

In contrast, RL operates directly on measured data and rewards from interaction with the environment. Reinforcement learning research has placed great focus on addressing cases which are analytically intractable by using approximations and data-driven techniques. An important class of approaches to RL within robotics centers on the use of classical optimal control techniques (e.g., linear-quadratic regulation and differential dynamic programming) to system models learned via repeated interaction with the environment [15.150, 153, 154]. A concise discussion of viewing RL as *adaptive optimal control* is presented in [15.155]. This path is indicated in Fig. 15.1 as *optimal control with learnt models*.

Reinforcement learning applied to robotics (Fig.15.6) differs considerably from most well-studied RL benchmark problems. Problems in robotics are

Avidbots Ex1016
Page 414 of 2154

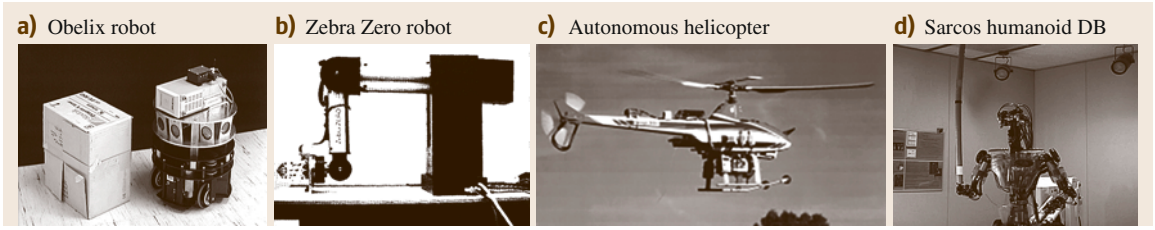


Fig.15.6a–d A small sample of robots with behaviors that were reinforcement learned. These cover the whole range of aerial vehicles, robotic arms, autonomous vehicles, and humanoid robots. **(a)** The OBELIX robot is a wheeled mobile robot that learned to push boxes (after [15.148]) with a value function-based approach (courtesy of Sridhar Mahadevan). **(b)** A Zebra Zero robot arm learned a peg-in-hole insertion task (after [15.149]) with a model-free policy gradient approach (courtesy of Rod Grupen). **(c)** Carnegie Mellon's autonomous helicopter leveraged a model-based policy search approach to learn a robust flight controller (after [15.150]). **(d)** The Sarcos humanoid DB learned a pole-balancing task (after [15.151]) using forward models

often involve high-dimensional, continuous states and actions (note that the 10–30-dimensional continuous actions common in robot RL are considered large [15.152]). In robotics, it is often unrealistic to assume that the true state is completely observable and noise free. The learning system will not be able to know precisely in which state it is and even vastly different states might look very similar. Thus, RL in robotics is often modeled as partially observed, a point we take up in detail in our formal model description below. The learning system must hence use filters to estimate the true state. It is often essential to maintain the information state of the environment that not only contains the raw observations but also a notion of uncertainty on its estimates (e.g., both the mean and the variance of a Kalman filter tracking the ball in the robot table tennis example).

Experience on a real physical system is tedious to obtain, expensive, and often hard to reproduce. Even getting to the same initial state is impossible for the robot table tennis system. Every single trial run, often called a roll-out or episode, is costly and, as a result, such applications force us to focus on difficulties that do not arise as frequently as in classical RL benchmark examples that are mostly simulation studies. In order to learn within a reasonable time frame, suitable approximations of state, policy, value function, and/or system dynamics need to be introduced. However, while real-world experience is costly, it usually cannot be replaced by learning in simulations alone. In analytical or learned models of the system, even small modeling errors can accumulate to a substantially different behavior. Hence, algorithms need to be robust with respect to models that do not capture all the details of the real system, also referred to as *under-modeling*, and to model uncertainty. Another challenge commonly faced in robot RL is the generation of appropriate reward functions. Rewards that guide the learning system quickly to success are needed to cope with the cost of real-world experience. This problem is called *reward shaping* [15.156] and represents a substantial manual contribution. Specifying good reward functions in robotics requires a fair amount of domain knowledge and may often be hard in practice.

Not every RL method is equally suitable for the robotics domain. In fact, many of the methods thus far demonstrated on difficult problems have been model-based [15.50, 132, 157] and robot learning systems often employ policy search methods rather than value function-based approaches [15.149, 150, 158–164]. Such design choices stand in contrast to possibly the bulk of the early research in the machine learning community [15.13, 165].

15.3.2 Reinforcement Learning Methods

In RL, an agent tries to optimize the accumulated reward over a time horizon, e.g., a manipulation task or even its entire lifetime. In an *episodic* setting, where the task is restarted at the end of each episode, the objective is to maximize the total reward per episode. If the task is on-going, a discounted accumulated reward which weighs earlier rewards with less influence is typically optimized. The agent is modeled as being in an environment with state $\boldsymbol{s} \in S$ and can perform actions $\boldsymbol{a} \in A$. The states and actions can be either discrete or continuous and are normally multi-dimensional. A state $\boldsymbol{s}$ contains all relevant information about the agent and environment needed to model future states or observables. An action $\boldsymbol{a}$ is used to control the time evolution of the system. For example, in a navigation task, the state could be the position and orientation of the robot, whereas the action corresponds to the torques applied to the wheels.

At every time step, the agent gets a reward R, a scalar value which typically depends on its state and

action. For instance, in a navigation task, rewards could be given for reaching the target, and penalties (negative reward) would arise for expending energy. The goal of RL is to find a policy π that chooses action $\boldsymbol{a}$ in $\boldsymbol{s}$ such as to maximize the cumulative reward over time. The policy π can be either deterministic, which always chooses the same action for each state $\boldsymbol{a} = \pi(\boldsymbol{s})$, or probabilistic which draws a sample action from a distribution: $\boldsymbol{a} \sim \pi(\boldsymbol{s}, \boldsymbol{a}) = P(\boldsymbol{a}|\boldsymbol{s})$. Reinforcement learning algorithms need to discover the relations between states, actions, and rewards. Hence exploration is required to sufficiently sample these functional relationships, which can be directly embedded in the policy or performed separately as part of the learning process.

Classical RL approaches are based upon the assumption of a *Markov decision process* or MDP which models the dynamics of the environment through transition probabilities T. Given the current state $\boldsymbol{s}$ and action $\boldsymbol{a}$, the conditional transition probability $T(\boldsymbol{s}', \boldsymbol{a}, \boldsymbol{s}) = P(\boldsymbol{s}'|\boldsymbol{s}, \boldsymbol{a})$ describes the distribution over the next state $\boldsymbol{s}'$ and generalize the notion of deterministic dynamics. The Markov property requires that the next state $\boldsymbol{s}'$ and reward depend only on the previous state $\boldsymbol{s}$ and action $\boldsymbol{a}$, and not on any other information [15.13]. In robotics, we are able to find an approximate notion of a state in a MDP for many situations.

Goals of Reinforcement Learning

The goal of RL is to discover an optimal policy π^* that maps states (or observations) to actions so as to maximize the expected return J, which corresponds to the cumulative expected reward. There are different models of optimal behavior [15.165] which result in different definitions of the expected return. A finite-horizon model only attempts to maximize the expected reward for the horizon H, i. e., the next H (time-)steps h

$$J = E\left\{\sum_{h=0}^{H} R_h\right\} .$$

This setting can also be applied to model problems where it is known how many steps are remaining.

Alternatively, future rewards can be discounted by a discount factor γ (with $0 \leq \gamma < 1$) yielding a metric

$$J = (1-\gamma)E\left\{\sum_{h=0}^{\infty} \gamma^h R_h\right\} .$$

This setting is most frequently discussed in classical RL texts. The parameter γ affects how much the future is taken into account and needs to be tuned manually. As illustrated in [15.165], this parameter often qualitatively changes the form of the optimal solution. Policies designed by optimizing with small γ are myopic and greedy, and may lead to poor performance if we actually care about longer term rewards. It is straightforward to show that the optimal control law can be unstable if the discount factor is too low (e.g., it is not difficult to show this destabilization even for discounted linear quadratic regulation problems). Hence, discounted formulations are occasionally inadmissible in robot control despite that they are somewhat more common in the machine learning community.

In the limit when the discount factor γ approaches 1, the metric approaches what is known as the average-reward criterion [15.7]

$$J = \lim_{H\to\infty} E\left\{\frac{1}{H}\sum_{h=0}^{H} R_h\right\} .$$

This setting has the problem that it cannot distinguish between policies that initially gain a transient of large rewards and those that do not. This transient phase, also called prefix, is dominated by the rewards obtained in the long run. If a policy accomplishes both an optimal prefix as well as an optimal long-term behavior, it is called bias optimal [15.166]. An example in robotics would be the transient phase during the start of a rhythmic movement, where many policies will accomplish the same long-term reward but differ substantially in the transient (e.g., there are many ways of starting the same gait in dynamic legged locomotion) allowing for room for improvement in practical application.

In real-world domains, the shortcomings of the discounted formulation are often more critical than those of the average reward setting as stable behavior is often more important than a good transient [15.167]. We also often encounter an episodic control task, where the task runs only for H time-steps and then reset (potentially by human intervention) and started over. This horizon H may be arbitrarily large, as long as the expected reward over the episode can be guaranteed to converge. As such episodic tasks are probably the most frequent ones, finite-horizon models are often the most relevant.

Two natural goals arise for the learner. In the first, we attempt to find an optimal strategy at the end of a phase of training or interaction. In the second, the goal is to maximize the reward over the whole time the robot is interacting with the world.

In contrast to supervised learning, the learner must first discover its environment and is not told the optimal action it needs to take. To gain information about the rewards and the behavior of the system, the agent needs to explore by considering previously unused actions or actions it is uncertain about. It needs to decide whether to play it safe and stick to well-known actions

Avidbots Ex1016
Page 416 of 2154

with (moderately) high rewards or to dare trying new things in order to discover new strategies with an even higher reward. This problem is commonly known as the *exploration–exploitation tradeoff*. However, as state spaces in robotics problems are often tremendously large due to continuous and high-dimensional state–action spaces, exploring the entire state–action space is mostly infeasible. This problem is often referred to as the *curse of dimensionality* [15.168].

Off-policy methods learn independently of the employed policy, i. e., an explorative strategy that is different from the desired final policy can be employed during the learning process. *On-policy* methods collect sample information about the environment using the current policy. As a result, exploration must be built into the policy and determines the speed of the policy improvements. Such exploration and the performance of the policy can result in an exploration–exploitation tradeoff between long- and short-term improvement of the policy. Modeling exploration models with probability distributions has surprising implications, e.g., stochastic policies have been shown to be the optimal stationary policies for selected problems [15.169, 170] and can even break the curse of dimensionality [15.171]. Furthermore, stochastic policies often allow the derivation of new policy update steps with surprising ease.

The agent needs to determine a correlation between actions and reward signals. An action taken does not have to have an immediate effect on the reward but can also influence a reward in the distant future. The difficulty in assigning credit for rewards is directly related to the horizon or mixing time of the problem. It also increases with the dimensionality of the actions as not all parts of the action may contribute equally.

15.3.3 Optimal Control with Learnt Models

The classical RL setup is a MDP where, additionally, to the states $\boldsymbol{s}$, actions $\boldsymbol{s}$, and rewards R we also have transition probabilities $T(\boldsymbol{s}', \boldsymbol{a}, \boldsymbol{s})$. Here, the reward is modeled as a reward function $R(\boldsymbol{s}, \boldsymbol{a})$. If both the transition probabilities and reward function are known, this can be seen as an optimal control problem [15.152]. If these functions are acquired from measured data through model learning methods as described in Sect. 15.2, we are in the scenario of *optimal control with learnt models* indicated in Fig. 15.1.

The value function $V^{\pi}(\boldsymbol{s})$ provides an assessment of much reward will be accumulated starting from state $\boldsymbol{s}$ and following a given policy π. If a policy is optimal, denoted as π^*, the corresponding optimal value function will be denoted a V^{π^*}, often abbreviated as V^*. For each state there can be one or several optimal actions $\boldsymbol{a}^*$, resulting in the *same* value $V^*(\boldsymbol{s})$. The consistency condition for optimality is

$$V^*(\boldsymbol{s}) = \max_{\boldsymbol{a}} \left[R(\boldsymbol{s}, \boldsymbol{a}) + \gamma \sum_{\boldsymbol{s}'} V^*(\boldsymbol{s}') T(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{s}') \right] \tag{15.11}$$

in the discounted case. This statement is equivalent to the *Bellman principle of optimality* [15.168] that states:

> *An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision.*

Thus, we have to perform an optimal action $\boldsymbol{a}^*$, and, subsequently, follow the optimal policy π^* in order to achieve a global optimum. This principle of optimality has also been crucial in enabling the field of optimal control [15.172]. Note that for constructing the optimal policy, the max operator in (15.11) needs to be evaluated, which requires knowledge of the state transition model $T(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{s}')$, i. e., this approach is generically a model-based requiring either a predefined model or a learnt one using insights from Sect. 15.2.

Dynamic programming-based methods are the classical approach for solving optimal control problems in RL. These require, as mentioned before, a model of the transition probabilities $T(\boldsymbol{s}', \boldsymbol{a}, \boldsymbol{s})$ and the reward function $R(\boldsymbol{s}, \boldsymbol{a})$ to calculate the value function. These functions do not necessarily need to be predetermined but can also be learned from data, potentially incrementally. Such methods are called *model-based* RL methods or optimal control with learnt models. Typical methods include policy iteration and value iteration.

Policy iteration alternates between the two phases of policy evaluation and policy improvement. The approach is initialized with an arbitrary policy. Policy evaluation determines the value function for the current policy. Each state is visited and its value is updated based on the current value estimates of its successor states, the associated transition probabilities, as well as the policy. This procedure is repeated until the value function converges to a fixed point, which approximates the true value function. Policy improvement greedily selects the best action in every state according to the value function, thus creating a policy update. The two steps of policy evaluation and policy improvement are iterated until the policy does not change any longer.

Policy iteration only updates the policy once the policy evaluation step has converged. In contrast, value iteration combines the steps of policy evaluation and policy improvement by directly updating the value function based on (15.11) every time a state is updated.

Avidbots Ex1016
Page 417 of 2154

15.3.4 Model-Free Value Function Approaches

Many traditional RL approaches are based on identifying (possibly approximate) solutions to this equation, and are known as *value function methods*. Instead of deriving the value functions from a transition model and a reward function, they first approximate the value function, and use it to construct the optimal policy.

Instead of the value function $V^{\pi}(\boldsymbol{s})$, these algorithms usually rely on the state–action value function $Q^{\pi}(\boldsymbol{s},\boldsymbol{a})$, which has advantages of determining the optimal policy, as shown below, but also enables the use of model-free methods. This state–action value function is defined as

$$Q^{\pi}(\boldsymbol{s},\boldsymbol{a}) = R(\boldsymbol{s},\boldsymbol{a}) + \gamma \sum_{s'} V^{\pi}(\boldsymbol{s}')T(\boldsymbol{s},\boldsymbol{a},\boldsymbol{s}') \,.$$

In contrast to the value function $V^{\pi}(\boldsymbol{s})$, the state–action value function $Q^{\pi}(\boldsymbol{s},\boldsymbol{a})$ explicitly contains the information about the effects of a particular action. The optimal state–action value function is

$$Q^{*}(\boldsymbol{s},\boldsymbol{a}) = R(\boldsymbol{s},\boldsymbol{a}) + \gamma \sum_{s'} \left(\max_{a'} Q^{*}(\boldsymbol{s}',\boldsymbol{a}') \right) T(\boldsymbol{s},\boldsymbol{a},\boldsymbol{s}') \,.$$

If Q^{*} is known, finding the optimal action just requires finding the maximal value of Q at a given state $\boldsymbol{s}$ – this does not require a model, but is just a standard (although not necessarily trivial) root-finding problem of functions. Importantly, using state–action value function allows a model-free learning approach, at the cost of learning a value function over the joint space of $\boldsymbol{s}$ and $\boldsymbol{a}$.

If the optimal value function $V^{*}(\boldsymbol{s}')$ and the transition probabilities $T(\boldsymbol{s},\boldsymbol{a},\boldsymbol{s}')$ for the following states are known, determining the optimal policy is straightforward in a setting with discrete actions, as an exhaustive search is possible. For continuous spaces, determining the optimal action $\boldsymbol{a}^{*}$ is an optimization problem in itself. If both states and actions are discrete, the value function and the policy may, in principle, be represented by tables and picking the appropriate action is reduced to a look-up. For large or continuous spaces, representing the value function as a table becomes intractable. Function approximation is employed to find a lower dimensional representation that matches the real value function as closely as possible, as discussed later.

A wide variety of methods of model-free value function based RL algorithms that attempt to estimate $V^{*}(\boldsymbol{s})$ or $Q^{*}(\boldsymbol{s},\boldsymbol{a})$ have been developed and can be split mainly into two classes: (i) rollout-based Monte Carlo methods and (ii) temporal difference methods such as TD(λ), Q-learning, and SARSA (state-action-reward-state-action).

Monte Carlo methods use sampling in order to estimate the value function. This procedure can be used to replace the policy evaluation step of the dynamic programming-based methods above. Monte Carlo methods are *model-free*, i. e., they do not need an explicit transition function. They perform roll-outs by executing the current policy on the system, hence operating on-policy. The frequencies of transitions and rewards are kept track of and are used to form estimates of the value function. For example, in an episodic setting the state–action value of a given state–action pair can be estimated by averaging all the returns that were received when starting from them.

Temporal difference methods, unlike Monte Carlo methods, do not have to wait until an estimate of the return is available (i. e., at the end of an episode) to update the value function. Rather, they use temporal errors and update at every time step. The temporal error is the difference between the old estimate and a new estimate of the value function, taking into account the reward received in the current sample. These updates are done iterativley and, in contrast to dynamic programming methods, only take into account the sampled successor states rather than the complete distributions over successor states. Like the Monte Carlo methods, these methods are model-free, as they do not use a model of the transition function to determine the value function. For example, the value function could be updated iteratively by

$$V'(\boldsymbol{s}) = V(\boldsymbol{s}) + \alpha \left[R(\boldsymbol{s},\boldsymbol{a}) + \gamma V(\boldsymbol{s}') - V(\boldsymbol{s}) \right] ,$$

where $V(\boldsymbol{s})$ is the old estimate of the value function, $V'(\boldsymbol{s})$ the updated one, and α is a learning rate. This update step is called the TD(0)-algorithm in the discounted reward case. However, in order to select the optimal action, a model of the transition function would still be required, i. e., this method can only be used for policy evaluation in a policy iteration scenario.

The equivalent temporal difference learning algorithm for state–action value functions is SARSA with

$$Q'(\boldsymbol{s},\boldsymbol{a}) = Q(\boldsymbol{s},\boldsymbol{a}) + \alpha \left[R(\boldsymbol{s},\boldsymbol{a}) + \gamma Q(\boldsymbol{s}',\boldsymbol{a}') - Q(\boldsymbol{s},\boldsymbol{a}) \right] ,$$

where $Q(\boldsymbol{s},\boldsymbol{a})$ is the old estimate of the state–action value function and $Q'(\boldsymbol{s},\boldsymbol{a})$ the updated one. This algorithm is on-policy as both the current action $\boldsymbol{a}$ as well as the subsequent action $\boldsymbol{a}'$ are chosen according to the current policy π. The average-reward variant is called R-learning [15.173], which is closely related to

Avidbots Ex1016
Page 418 of 2154

Q-learning, with the updates

$$Q'(\boldsymbol{s},\boldsymbol{a}) = Q(\boldsymbol{s},\boldsymbol{a}) + \alpha\left[R(\boldsymbol{s},\boldsymbol{a}) + \gamma \max_{\boldsymbol{a}'} Q(\boldsymbol{s}',\boldsymbol{a}') - Q(\boldsymbol{s},\boldsymbol{a})\right].$$

These methods do not require a model of the transition function for determining the deterministic optimal policy $\pi^*(\boldsymbol{s})$.

15.3.5 Policy Search

Value function approaches infer the optimal policy indirectly by first finding a value function, and then computing the corresponding optimal actions. Instead, the policy can be learned directly, without a detour through a value function. For robotics, this approach has several advantages. It allows for a natural integration of expert knowledge, e.g., through both structure and initializations of the policy. It allows domain-appropriate prestructuring of the policy in an approximate form without changing the original problem. Optimal policies often have many fewer parameters than optimal value functions. For example, in linear quadratic control, the value function has quadratically many parameters in the dimensionality of the state-variables, while the policy requires only linearly many parameters. Local search in policy space can directly lead to good results as exhibited by early hill-climbing approaches [15.172], as well as more recent successes, [15.149, 150, 157, 159–163, 174–182]. Additional constraints can be incorporated naturally, e.g., regularizing the change in the path distribution. As a result, policy search often appears more natural to robotics.

Policy search has been considered the harder problem for a long time as the optimal solution most often cannot directly be determined from equations as in *Bellman principle of optimality* [15.168] and dynamic programming. Nevertheless, in robotics, policy search has recently become an important alternative to value function based methods due to better scalability as well as the convergence problems of approximate value function methods. Most policy search methods optimize locally around existing policies π, parameterized by a set of policy parameters θ_i, by computing changes in the policy parameters $\Delta\theta_i$ that will increase the expected return and results in iterative updates of the form

$$\theta_{i+1} = \theta_i + \Delta\theta_i\,.$$

The computation of the policy update is the key step here and a variety of updates have been proposed, including pairwise comparisons [15.183, 184], gradient estimation using finite policy differences [15.159, 160, 180, 185–187], general stochastic optimization methods (such as Nelder-Mead [15.150], cross entropy [15.188] and population-based methods [15.189]), and approaches coming from optimal control such as differential dynamic programming or DDP [15.153] and multiple shooting approaches [15.190].

Let us briefly take a closer look at gradient-based approaches first. The updates of the policy parameters are based on a hill-climbing approach, which is following the gradient of the expected return J for a defined step-size α

$$\theta_{i+1} = \theta_i + \alpha\nabla_\theta J\,.$$

Different methods exist for estimating the gradient $\nabla_\theta J$ and many algorithms require tuning of the step-size α.

Finite difference gradient estimators evaluate P policy parameter perturbations to obtain an estimate of the policy gradient. Here, we have $\Delta\hat{J}_p \approx J(\theta_i + \Delta\theta_p) - J_{\text{ref}}$, where $p = [1,\ldots,P]$ are the individual perturbations, $\Delta\hat{J}_p$ the estimate of their influence on the return, and J_{ref} is a reference return, e.g., the return of the unperturbed parameters. The gradient can now be estimated by linear regression

$$\nabla_\theta J \approx \left(\Delta\boldsymbol{\Theta}^{\mathrm{T}}\Delta\boldsymbol{\Theta}\right)^{-1}\Delta\boldsymbol{\Theta}^{\mathrm{T}}\Delta\hat{\mathbf{J}}\,,$$

where the matrix $\Delta\boldsymbol{\Theta}$ contains all the stacked samples of the perturbations $\Delta\theta_p$ and $\Delta\hat{\mathbf{J}}$ contains the corresponding $\Delta\hat{J}_p$. In order to estimate the gradient the number of perturbations needs to be at least as large as the number of parameters. The approach is very straightforward and even applicable to policies that are not strictly differentiable. However, it is usually considered to be very noisy and inefficient. For the finite difference approach tuning the step-size α for the update, the number of perturbations P, and the type and magnitude of perturbations are all critical tuning factors.

Likelihood ratio methods rely on the insight that in an episodic setting where the episodes τ are generated according to the distribution $P^\theta(\tau) = P(\tau|\theta)$ with the return of an episode $J^\tau = \sum_{h=1}^{H} R_h$ and number of steps H in each episode, the expected return for a set of policy parameter θ can be expressed as

$$J^\theta = \sum_\tau P^\theta(\tau)J^\tau\,. \tag{15.12}$$

The gradient of the episode distribution can be written as

$$\nabla_\theta P^\theta(\tau) = P^\theta(\tau)\nabla_\theta \log P^\theta(\tau)\,, \tag{15.13}$$

which is commonly known as the likelihood ratio or *REINFORCE* (reward increment = nonnegative factor × offset reinforcement × characteristic eligibility) [15.191] trick. From multivariate calculus, we have $\nabla_\theta \log P^\theta(\tau) = \nabla_\theta P^\theta(\tau)/P^\theta(\tau)$. Combining (15.12) and (15.13) we get the gradient of the expected return in the form

$$\begin{aligned}\nabla_\theta J^\theta &= \sum_\tau \nabla_\theta P^\theta(\tau) J^\tau \\ &= \sum_\tau P^\theta(\tau) \nabla_\theta \log P^\theta(\tau) J^\tau \\ &= E\left\{\nabla_\theta \log P^\theta(\tau) J^\tau\right\}.\end{aligned}$$

If we have a stochastic policy $\pi^\theta(\boldsymbol{a}|\boldsymbol{s})$ that generates the episodes τ, we do not need to keep track of the probabilities of the episodes but can directly express the gradient in terms of the policy as $\nabla_\theta \log P^\theta(\tau) = \sum_{h=1}^H \nabla_\theta \log \pi^\theta(\boldsymbol{a}_h|\boldsymbol{s}_h)$. Finally the gradient of the expected return with respect to the policy parameters can be estimated as

$$\nabla_\theta J^\theta = E\left\{\left(\sum_{h=1}^H \nabla_\theta \log \pi^\theta(\boldsymbol{a}_h|\boldsymbol{s}_h)\right) J^\tau\right\}.$$

If we now take into account that rewards at the beginning of an episode cannot be caused by actions taken at the end of an episode, we can replace the return of the episode J^τ by the state–action value function $Q^\pi(\boldsymbol{s}, \boldsymbol{a})$ and get [15.161]

$$\begin{aligned}\nabla_\theta J^\theta = (1-\gamma) \\ \times E\left\{\sum_{h=1}^H \gamma^h \nabla_\theta \log \pi^\theta(\boldsymbol{a}_h|\boldsymbol{s}_h) Q^\pi(\boldsymbol{s}_h, \boldsymbol{a}_h)\right\},\end{aligned}$$

which is equivalent to the *policy gradient theorem* [15.169]. In practice, it is often advisable to subtract a reference J_{ref}, also called baseline, from the return of the episode J^τ or the state–action value function $Q^\pi(s, a)$, respectively, to get better estimates, similar to the finite difference approach. In these settings, the exploration is automatically taken care of by the stochastic policy.

Initial gradient-based approaches such as finite differences gradients or REINFORCE [15.191] have been rather slow. The weight perturbation algorithm is related to REINFORCE but can deal with non-Gaussian distributions which significantly improves the signal-to-noise ratio of the gradient [15.180]. Recent natural policy gradient approaches [15.161, 162] have allowed for faster convergence which may be advantageous for robotics as it reduces the learning time and required real-world interactions.

Alternatives to gradient methods are currently a hot research topic. For example, a different class of safe and fast policy search methods, inspired by expectation-maximization from machine learning, can be derived when the reward is treated as an improper probability distribution [15.192]. Some of these approaches have proven successful in robotics, e.g., reward-weighted regression [15.161], policy learning by weighting exploration with the returns [15.163], Monte Carlo expectation maximization [15.193], and cost-regularized kernel regression [15.194]. Algorithms with closely related update rules can also be derived from different perspectives including policy improvements with path integrals [15.195] and relative entropy policy search [15.175].

Finally, the policy search by dynamic programming [15.196] method is a general strategy that combines policy search with the principle of optimality. The approach learns a nonstationary policy backward in time like dynamic programming methods, but does not attempt to enforce the Bellman equation and the resulting approximation instabilities. The resulting approach provides some of the strongest guarantees that are currently known under function approximation and limited observability. It has been demonstrated in learning walking controllers and in finding near-optimal trajectories for map exploration [15.197].

One of the key open issues in the field is determining when it is appropriate to use each of these methods. Some approaches leverage significant structure specific to the RL problem [15.195], including reward structure, Markovanity, causality of reward signals [15.191], and value-function estimates when available [15.161]. Others embed policy search as a generic, black-box, problem of stochastic optimization [15.150, 198–201]. Significant open questions remain regarding which methods are best in which circumstances and further, at an even more basic level, how effective leveraging the kinds of problem structures mentioned above are in practice. Examples of resulting success can be observed in VIDEO 354 and VIDEO 355.

Value Function Approaches Versus Policy Search

The discussion whether value function approaches or policy search methods are more suitable for RL has been going on for a long time, and, so far, the answer is really problem specific. If a complete optimal value function is known, a globally optimal solution follows by greedily choosing actions that receive the highest value. However, value-function-based approaches have up to now been difficult to scale to high-dimensional

Avidbots Ex1016
Page 420 of 2154

robotics as they require function approximation for the value function – in low dimensions, discretizations and lookup tables work best and have no theoretical issues. Unfortunately, theoretical guarantees from discrete state–action representations no longer hold when employing function approximation. Even finding the optimal action can be a hard problem due to the sensitivity of selecting the optimal action in face of even tiny inaccuracies in the approximated value function, and the cost of solving an optimization problem in a continuous unknown function for action selection. For example, the computational cost of updating a policy in policy search can easily be cheaper than finding just *one* optimal action for one state by searching the state–action value function.

In principle, a value function requires total coverage over the state space and the *largest local error* determines the quality of the resulting policy. A particularly significant problem is the error propagation in value functions. A small change in the policy may cause a large change in the value function, which again causes a large change in the policy. While this may lead more quickly to good, possibly globally optimal solutions, such learning processes often prove unstable under function approximation [15.196, 202, 203] and are considerably more dangerous when applied to real robotic systems where overly large policy deviations may lead to dangerous motor commands.

In contrast, policy search methods usually only consider the current policy and its neighborhood in order to gradually improve performance. The result is that usually only local optima, and not the global one, can be found. However, these methods work well in conjunction with continuous states and actions, and they scale to high-dimensional robots, e.g., humanoid robots.

Policy search methods are sometimes called *actor*-only methods; value function methods are sometimes called *critic*-only methods. The idea of a critic is to first observe and estimate the performance of choosing controls on the system (i. e., the value function), then derive a policy based on the gained knowledge. In contrast, the actor directly tries to deduce the optimal policy. A set of algorithms called *actor-critic* methods attempt to incorporate the advantages of each: a policy is explicitly maintained, as is a value function for the current policy. The value function (i. e., the critic) is not employed for action selection. Instead, it observes the performance of the actor and decides when the policy needs to be updated and which action should be preferred. The resulting update step features the local convergence properties of policy gradient algorithms while reducing update variance [15.204]. There is a tradeoff between the benefit of reducing the variance of the updates and having to learn a value function as the samples required to estimate the value function could also be employed to obtain better gradient estimates for the update step. *Rosenstein* and *Barto* [15.205] propose an actor-critic method that additionally features a supervisor in the form of a stable policy.

Function Approximation

The topic of *function approximation* occurs repeatedly in RL and deserves some attention. For discrete state–action problems with not too many states and actions, look-up tables are a reasonable representation for value functions, rewards functions, and state-transitions (i. e., models). But look-up tables have no generalization to neighboring states, which makes them computationally inefficient. Moreover, for continuous state–action problems with many dimensions, discretization would create an exponential growth of the number of discrete states and actions, and is thus infeasible (*curse of dimensionality*). Thus, replacing discretized representations by continuous function approximators is the only possibility.

Function approximation can be employed to represent policies, value functions, and state-transition models. Thus, the techniques discussed in Sect. 15.2.4 also apply to these function approximation problems. However, a fundamental problem in using supervised learning methods developed in the literature for function approximation is that most such methods are designed for independently and identically distributed sample data. However, the data generated by the RL process is usually neither independent nor identically distributed. Usually, the function approximator itself plays some role in the data collection process (e.g., by serving to define a policy that we execute on a robot.)

Linear basis function approximators form one of the most widely used approximate value function techniques in continuous (and discrete) state spaces. This popularity is largely due to the simplicity of their representation as well as a convergence theory, albeit limited, for the approximation of value functions based on samples [15.206]. Let us briefly take a closer look at a radial basis function network to illustrate this approach. The value function maps states to a scalar value. The state space can be covered by a grid of points, each of which correspond to the center of a Gaussian-shaped basis function. The value of the approximated function is the weighted sum of the values of all basis functions at the query point. As the influence of the Gaussian basis functions drops rapidly with the distance of its center, the value of the query points will be predominantly influenced by the neighboring basis functions. The weights are set in a way to minimize the error between the observed samples and the reconstruction. For the mean squared error, these weights can be de-

Avidbots Ex1016
Page 421 of 2154

termined by linear regression. *Kolter* and *Ng* [15.207] discuss the benefits of regularization of such linear function approximators to avoid over-fitting.

Other possible function approximators for value functions include wire fitting, which *Baird* and *Klopf* [15.208] suggested as an approach that makes continuous action selection feasible. Using Fourier basis has been suggested by *Konidaris* et al. [15.209]. Even discretizing the state space can be seen as a form of function approximation where coarse values serve as estimates for a smooth continuous function. One example is tile coding [15.13], where the space is subdivided into (potentially irregularly shaped) regions, called tiling. The number of different tilings determines the resolution of the final approximation. Policy search also benefits from compact representations of the policy.

Models of the system dynamics can be represented using a wide variety of techniques. In this case, it is often important to model the uncertainty in the model (e.g., by a stochastic model or Bayesian estimates of model parameters) to ensure that the learning algorithm does not exploit model inaccuracies. See Sect. 15.3.9 for a more detailed discussion.

15.3.6 Challenges in Robot Reinforcement Learning

Reinforcement learning is generally a difficult problem and many of its challenges are particularly apparent in the robotics setting. As the states and actions of most robots are inherently continuous, we are forced to consider the resolution at which they are represented. We must decide how fine grained the control is that we require over the robot, whether we employ discretization or function approximation, and what time step we establish. Additionally, as the dimensionality of both states and actions can be high, we face the *Curse of Dimensionality* [15.168] as discussed in Sect. 15.3.6, *Curse of Dimensionality*. As robotics deals with complex physical systems, samples can be expensive due to the long execution time of complete tasks, required manual interventions, and the need for maintenance and repair. In these real-world measurements, we must cope with the uncertainty inherent in complex physical systems. A robot requires that the algorithm runs in real-time. The algorithm must be capable of dealing with delays in sensing and execution that are inherent in physical systems (Sect. 15.3.6). A simulation might alleviate many problems but these approaches need to be robust with respect to model errors as discussed later. An often underestimated problem is the goal specification, which is achieved by designing a good reward function. As noted in Sect. 15.3.6, this choice can make the difference between feasibility and an unreasonable amount of exploration.

Curse of Dimensionality

When *Bellmann* [15.168] explored optimal control in discrete high-dimensional spaces, he faced an exponential explosion of states and actions for which he coined the term *Curse of Dimensionality*. As the number of dimensions grows, exponentially more data and computation is needed to cover the complete state–action space. Evaluating every state quickly becomes infeasible with growing dimensionality, even for discrete states.

Robotic systems often have to deal with these high-dimensional states and actions due to the many degrees of freedom of modern anthropomorphic robots. For example, in a ball-paddling task (VIDEO 355) the proper representation of a robot's state would consist of its joint angles and velocities for each of its seven degrees of freedom as well as the Cartesian position and velocity of the ball (Fig. 15.7). The robot's actions would be the generated motor commands, which often are torques or accelerations. In this example, we have $2 \times (7+3) = 20$ state dimensions and seven-dimensional continuous actions.

In robotics, such tasks are often rendered tractable to the robot engineer by a hierarchical task decomposition that shifts some complexity to a lower layer of functionality. Classical RL approaches often consider a grid-based representation with discrete states and actions, often referred to as a *grid-world*. A navigational task for mobile robots could be projected into

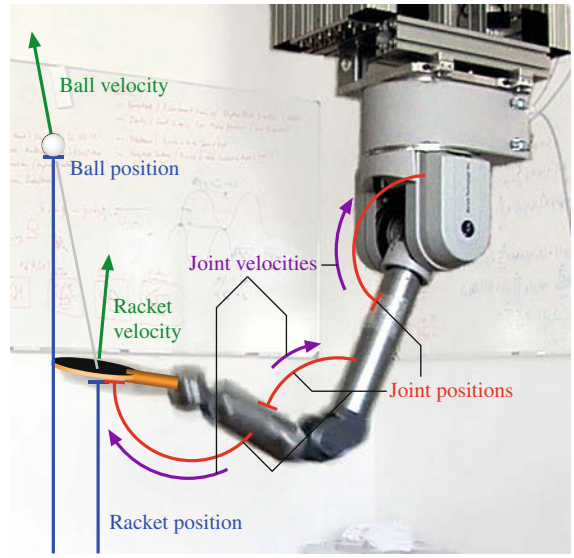


Fig. 15.7 State–space used in the modeling of a robot RL task of paddling a ball. This example can also be observed in VIDEO 355

Avidbots Ex1016
Page 422 of 2154

this representation by employing a number of actions like *move to the cell to the left* that use a lower level controller that takes care of accelerating, moving, and stopping while ensuring precision. In the ball-paddling example, we may simplify by controlling the robot in racket space (which is lower dimensional as the racket is orientation-invariant around the string's mounting point) with an operational space control law [15.15]. Many commercial robot systems also encapsulate some of the state and action components in an embedded control system (e.g., trajectory fragments are frequently used as actions for industrial robots). However, this form of a state dimensionality reduction severely limits the dynamic capabilities of the robot according to our experience [15.63, 210].

The RL community has a long history of dealing with dimensionality using computational abstractions. It offers a larger set of applicable tools ranging from adaptive discretizations [15.211] and function approximation approaches [15.13] to macro-actions or options [15.212, 213]. Options allow a task to be decomposed into elementary components and quite naturally translate to robotics. Such options can autonomously achieve a subtask, such as opening a door, which reduces the planning horizon [15.212]. The automatic generation of such sets of options is a key issue in order to enable such approaches. We will discuss approaches that have been successful in robot RL in Sect. 15.3.7.

Curse of Real-World Samples

Robots inherently interact with the physical world and, thus, create their own learning data from physical sensors. Real physical systems create a variety of problems that are worthwhile to keep in mind:

- Robot hardware is usually expensive, suffers from wear and tear, and requires careful maintenance. Repairing a robot system is a nonnegligible effort associated with cost, physical labor, and long waiting periods. To apply RL in robotics, safe exploration becomes a key issue of the learning process [15.157, 214–216].
- The robot and its environment may continually change slowly, e.g., due to wear and tear, or light conditions. Thus, the learning process should be able to track these changing conditions, i. e., learn continually.
- Experimentation with real robots is time consuming, e.g., putting the pole back on the robot's end-effector during pole balancing after a failure, and running many trials for learning. Algorithms that are data efficient are often more important than memory and computations needed to process the data.
- Real-time constraints on an actual robot impose constraints for how much computation can be performed for action generation and learning updates. These constraints are less severe in an episodic setting where the time intensive part of the learning can be postponed to the period between episodes. *Hester* et al. [15.217] have proposed a real-time architecture for model-based value function RL methods taking into account these challenges.
- Time-discretization, inevitable in computer implementations, can generate undesirable artifacts, e.g., the distortion of distance between states, or smoothing effect by temporal aliasing.
- Delays in information processing with physical sensors and actuators can negatively affect learning. This effect can be addressed by putting some number of recent actions into the state, at the cost of significantly increasing the dimensionality of the problem.

Curse of Under-Modeling and Model Uncertainty

One way to offset the cost of real-world interaction is to use accurate models as simulators as suggested in Sect. 15.2.5. In an ideal setting, this approach would render it possible to learn the behavior in simulation and subsequently transfer it to the real robot. Unfortunately, creating a sufficiently accurate model of the robot and its environment is often impossible. As small modeling errors accumulate over time, the simulated robot can quickly diverge from the real-world system, such that the policy will not transfer without significant modifications as experienced by [15.218]. For tasks where the system is self-stabilizing (i. e., where the robot does not require active control to remain in a safe state or return to it), transferring policies often works much better [15.219]. Learning in simulation, however, becomes quickly useless when contact dynamics and friction are too hard to model. For example, in the ball-paddling task the elastic string that attaches the ball to the racket, and the contact dynamics of the bouncing ball have such modeling problems.

Curse of Goal Specification

In RL, the desired behavior is implicitly specified by the reward function. While often dramatically simpler than specifying the behavior itself, in practice, it can be surprisingly difficult to define a good reward function, such that manual design and tuning become very important. In many domains, it seems natural to provide rewards only upon task achievement – for example, when a table tennis robot wins a match. This view results in an apparently simple, binary reward specification. However, a robot may receive such a reward so rarely that it is un-

likely to ever succeed in its lifetime. Instead of relying on simpler binary rewards, we frequently need to include intermediate rewards in the scalar reward function to guide the learning process to a reasonable solution, a process known as *reward shaping* [15.156]. Additionally, physical constraints of the robot in terms of range of motion and torque saturation need to be taken into account. Often, a change in coordinates, i. e., the question in which space (e.g., task space, joint space) the cost function should be specified, can determine success or failure. In general, RL algorithms are also notorious for exploiting the reward function in ways that are not anticipated by the designer.

Inverse RL, also known as inverse optimal control [15.220], is a promising alternative to specifying the reward function manually. It assumes that a reward function can be reconstructed from a set of expert demonstrations as indicated in Fig. 15.1. This reward function does not necessarily correspond to the true reward function, but provides guarantees on the resulting performance of learned behaviors [15.221, 222]. Inverse RL was initially studied in the control community [15.8] and in the field of economics [15.223]. The initial results were only applicable to limited domains (linear quadratic regulator problems) and required closed form access to plant and controller, hence samples from human demonstrations could not be used. *Russel* [15.220] brought the field to the attention of the machine learning community. *Abbeel* and *Ng* [15.221] defined an important constraint on the solution to the inverse RL problem when reward functions are linear in a set of features: a policy that is extracted by observing demonstrations has to earn the same reward as the policy that is being demonstrated. *Ratliff* et al. [15.222] demonstrated that inverse optimal control can be understood as a generalization of ideas in machine learning of *structured prediction* and introduced efficient subgradient-based algorithms with regret bounds that enabled large scale application of the technique within robotics. *Ziebart* et al. [15.2] extended the technique developed by *Abbeel* and *Ng* [15.221] by rendering the idea robust and probabilistic, enabling its effective use for both learning policies and predicting the behavior of suboptimal agents. These techniques, and many variants, have been recently successfully applied to outdoor robot navigation [15.224–226], manipulation [15.227], and quadruped locomotion [15.224, 227, 228]. An example of a resulting success can be observed in VIDEO 353.

More recently, the notion that complex policies can be built on top of simple, easily solved optimal control problems by exploiting rich, parametrized reward functions has been exploited within RL more directly. In [15.229, 230], complex policies are derived by adapting a reward function for simple optimal control problems using policy search techniques. *Zucker* and *Bagnell* [15.230] demonstrate that this technique can enable efficient solutions to robotic marble-maze problems that effectively transfer between mazes of varying design and complexity. These works highlight the natural tradeoff between the complexity of the reward function and the complexity of the underlying RL problem for achieving a desired behavior.

15.3.7 Tractability Through Representation

Much of the success of RL methods has been due to the clever use of approximate representations. The need of such approximations is particularly pronounced in robotics, where table-based representations (as discussed in Sect. 15.3.4) are rarely scalable. Common themes are to reduce the dimensionality of states and actions, to choose representations that are robust in face of the data generation process of RL, or to find compact parameterizations of policies and value functions. The following outlines a list of items that are relevant:

- A variety of authors have manually developed discretizations so that basic tasks can be learned on real robots. For low-dimensional tasks, we can generate discretizations straightforwardly by splitting each dimension into a number of regions. The main challenge is to find the right number of regions for each dimension that allows the system to achieve a good final performance while still learning quickly. Example applications include balancing a ball on a beam [15.231], one degree of freedom ball-in-a-cup [15.232], two degree of freedom crawling motions [15.233], and gait patterns for four-legged walking [15.234]. Complex tasks need much more hand crafting, e.g., as in navigation [15.235], vision-based processing [15.236], or RoboCup scenarios [15.237].
- Adaptive representations can be very useful, e.g., as in cooperative task achievement [15.238], or computer vision [15.239].
- Automatic construction of meta-actions (and the closely related concept of options) has fascinated RL researchers. The idea is to have more intelligent actions that are composed of a sequence of commands and that in themselves achieve a simple task. A simple example would be to have a meta-action *move forward* 5 m. For example, in [15.240], the state and action sets are constructed in such a way that repeated action primitives lead to a change in the state to overcome problems associated with the discretization. *Huber* and *Grupen* [15.241] use a set of controllers with associated predicate states as

Avidbots Ex1016
Page 424 of 2154

a basis for learning turning gates with a quadruped. *Fidelman* and *Stone* [15.242] use a policy search approach to learn a small set of parameters that controls the transition between a walking and a capturing meta-action in a RoboCup scenario. A task of transporting a ball with a dog robot [15.243] can be learned with semiautomatically discovered options. Using only the subgoals of primitive motions, a humanoid robot can learn a pouring task [15.244]. Other examples include foraging [15.245] and cooperative tasks [15.246] with multiple robots, grasping with restricted search spaces [15.247], and mobile robot navigation [15.248]. If the meta-actions are not fixed in advance, but rather learned at the same time, these approaches are hierarchical RL approaches. *Konidaris* et al. [15.249, 250] propose an approach that constructs a skill tree from human demonstrations. Here, the skills correspond to options and are chained to learn a mobile manipulation skill.

- In a relational representation, the states, actions, and transitions are not represented individually. Entities of the same predefined type are grouped and their relationships are considered. This representation may be preferable for highly geometric tasks (which frequently appear in robotics) and has been employed to learn to navigate buildings with a real robot in a supervised setting [15.251] and to manipulate articulated objects in simulation [15.252].
- Function approximation has always been the key component that allowed value function methods to scale into interesting domains. Unfortunately, the max-operator used within the Bellman equation and temporal-difference updates can theoretically make most linear or nonlinear approximation schemes unstable for either value iteration or policy iteration. Quite frequently such an unstable behavior is also exhibited in practice. Linear function approximators are stable for policy evaluation, while nonlinear function approximation (e.g., neural networks) can even diverge if just used for policy evaluation [15.206]. A large number of approaches have been suggested to facilitate value function approximation. For instance, creating physics-inspired features as nonlinear basis functions for linear approximator [15.151, 253] can be promising. Generic neural networks can be successful [15.254, 255], in particular the CMAC (cerebellar model articulation controller) neural network [15.256]. Local model function approximators often provide more stable learning behavior [15.151, 257, 258]. Probabilistic nonparametric function approximators like Gaussian process regression has recently shown interesting successes in robot RL [15.259–262].
- Similar to choosing a good structure in value function approximation, choosing a good structure and parameterization for the policy can make a big different for success or failure or RL. Finding sparse parameterization essentially reduces the dimensionality of the learning problem, as only a few parameters need to be learned, and not an action for every time step. Among the most popular have been spline-based parameterizations [15.158, 180, 199], linear parameterizations based on physics-inspired features [15.263], parameterized motor primitives [15.161, 162, 264, 265], Gaussian Mixture Models [15.157, 177, 266], generic neural networks [15.149, 176, 185], and nonparametric approaches [15.175, 194].

15.3.8 Tractability Through Prior Knowledge

Prior knowledge can dramatically help guide the learning process. It can be included in the form of initial policies, demonstrations, initial models, a predefined task structure, or constraints on the policy such as torque limits or ordering constraints of the policy parameters. These approaches significantly reduce the search space and, thus, speed up the learning process. Providing a (partially) successful initial policy allows a RL method to focus on promising regions in the value function or in policy space. Prestructuring a complex task such that it can be broken down into several more tractable ones can significantly reduce the complexity of the learning task. Constraints may also limit the search space, but often pose new, additional problems for the learning methods. For example, policy search limits often do not handle hard limits on the policy well. Relaxing such constraints (a trick often applied in machine learning) is not feasible if they were introduced to protect the robot in the first place.

As indicated in Fig. 15.1, observed expert data can be used for constructing useful policies. Using demonstrations to initialize RL has become rather popular and provides multiple benefits. Perhaps the most obvious benefit is that it provides supervised training data of what actions to perform in states that are encountered. Such data may be helpful when used to bias policy action selection. The most dramatic benefit, however, is that demonstration – or a hand-crafted initial policy – removes the need for *global exploration* of the policy or state-space of the RL problem. The student can improve by locally optimizing a policy knowing what states are important, making local optimization methods feasible. Of course, if the demonstration was not close to the globally optimal behavior, only local optima can be discovered.

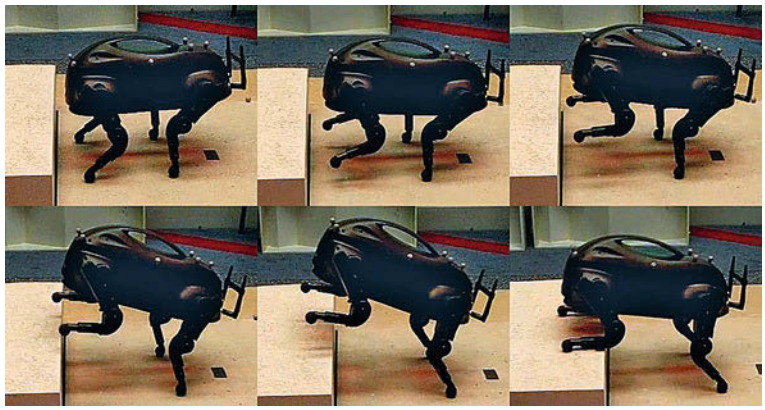


Fig. 15.8 Boston Dynamics LittleDog jumping (after [15.178], courtesy of Zico Kolter)

Demonstrations by a teacher can be obtained in two different scenarios. In the first, the teacher demonstrates the task using his or her own body; in the second, the teacher controls the robot to do the task. The first scenario is limited by the embodiment issue, as the movement of a human teacher usually cannot be mapped directly to the robot due to different physical constraints and capabilities. For example, joint angles of a human demonstrator need to be adapted to account for the kinematic differences between the teacher and the robot. Often it is more advisable to only consider task-relevant information, such as the Cartesian positions and velocities of the end-effector and the object. Demonstrations obtained by motion capture have been used to learn a pendulum swingup [15.174], ball-in-a-cup [15.267], and grasping [15.260].

The second scenario obtains demonstrations by a human teacher directly controlling the robot. Here the human teacher first has to learn how to achieve a task with the particular robot's hardware, adding valuable prior knowledge. If the robot is back-drivable, kinesthetic teach-in (i. e., by taking it by the hand and moving it) can be employed, which enables the teacher to interact more closely with the robot. This method has resulted in applications including T-ball batting [15.161, 162], reaching tasks [15.177, 268], ball-in-a-cup [15.163] and VIDEO 355, flipping a light switch [15.269], playing pool and manipulating a box [15.270], and opening a door and picking up objects [15.271]. A marble maze task can be learned using demonstrations by a human player [15.272].

Often a task can be decomposed hierarchically into basic components or into a sequence of increasingly difficult tasks. In both cases the complexity of the learning task is significantly reduced. For example, hierarchical Q-learning has been used to learn different behavioral levels for a six-legged robot: moving single legs, locally moving the complete body, and globally moving the robot toward a goal [15.273]. A stand-up behavior considered as a hierarchical RL task has been learned using Q-learning in the upper-level and a continuous actor-critic method in the lower level [15.274]. Navigation in a maze can be learned using an actor-critic architecture by tuning the influence of different control modules and learning these modules [15.275]. *Huber* and *Grupen* [15.241] combine discrete event system and RL techniques to learn turning gates for a quadruped. *Hart* and *Grupen* [15.213] learned to bi-manual manipulation tasks by assembling policies hierarchically. *Daniel* et al. [15.276] learn options in a tetherball scenario and *Muelling* et al. [15.147] learn different strokes in a table tennis scenario (VIDEO 354). *Whitman* and *Atkeson* [15.277] show that the optimal policy for some global systems (like a walking controller) can be constructed by finding the optimal controllers for simpler subsystems and coordinating these.

As discussed in Sect. 15.3.2, balancing exploration and exploitation is an important consideration. Task knowledge can be employed to guide to robots curiosity to focus on regions that are novel and promising at the same time. For example, a mobile robot learns to direct attention by employing a modified Q-learning approach using novelty [15.278]. Using *corrected truncated returns* and taking into account the estimator variance, a six-legged robot employed with stepping reflexes can learn to walk [15.279]. Offline search can be used to guide Q-learning during a grasping task [15.280]. Using upper confidence bounds [15.281] to direct exploration into regions with potentially high rewards, grasping can be learned efficiently [15.262].

15.3.9 Tractability Through Models

In Sect. 15.3.2, we discussed robot RL from a model-free perspective where the system simply served as a data generating process. Such model-free reinforce-

Avidbots Ex1016
Page 426 of 2154

ment algorithms try to directly learn the value function or the policy without any explicit modeling of the transition dynamics. In contrast, many robot RL problems can be made tractable by learning forward models, i. e., approximations of the transition dynamics based on data. Such model-based RL approaches jointly learn a model of the system with the value function or the policy and often allow for training with less interaction with the the real environment. Reduced learning on the real robot is highly desirable as simulations are frequently faster than real-time while safer for both the robot and its environment. The idea of combining learning in simulation and in the real environment was popularized by the Dyna-architecture [15.282], prioritized sweeping [15.283], and incremental multi-step Q-learning [15.284] in RL. In robot RL, the learning step on the simulated system is often called *mental rehearsal*.

A few remarks concerning model-based RL are important. It is essentially impossible to obtain a forward model that is accurate enough to simulate a complex real-world robot system without error. Reinforcement learning approaches exploit such model inaccuracies if they are beneficial for the reward received in simulation [15.174]. The resulting policies may work well with the forward model (i. e., the simulator) but poorly on the real system. This effect is known as simulation bias.

Nevertheless, simulation biases can be addressed by introducing stochastic models or distributions over models even if the system is very close to deterministic. Artificially adding a little noise will smooth model errors and avoid policy over-fitting [15.153, 285]. Model learning methods that maintain probabilistic uncertainty about true system dynamics allow the RL algorithm to generate distributions over the performance of a policy. Such methods explicitly model the uncertainty associated with the dynamics at each state and action. For example, when using a Gaussian process model of the transition dynamics, a policy can be evaluated by propagating the state and associated uncertainty forward in time. Such evaluations in the model can be used by a policy search approach to identify where to collect more data to improve a policy, and may be exploited to ensure that control is safe and robust to model uncertainty [15.150, 214]. When the new policy is evaluated on the real system, the novel observations can subsequently be incorporated into the forward model. *Bagnell* and *Schneider* [15.150] showed that maintaining model uncertainty and using it in the inner-loop of a policy search method enabled effective flight control using only minutes of collected data, while performance was compromised by considering a best-fit model. This approach uses explicit Monte Carlo simulation in the sample estimates. By treating model uncertainty as if it were noise [15.214] as well as employing analytic approximations of forward simulation, a cart-pole task can be solved with less than 20 s of interaction with the physical system [15.157]; a visually driven block-stacking task has also been learned data-efficiently [15.164].

15.4 Conclusions

We focussed on the two key branches of robot learning, i. e., supervised learning of internal models with regression methods, and trial-and-error with reinforcement learning. While both topics are traditional topics of machine learning, inserting these learning topics into *robot learning* adds a variety of complexities and intricacies, which are not immediately obvious – indeed many issues have been revealed only through experiments with physical robots. A large number of issues are control theoretic, i. e., due to dealing with a closed-loop physical system that needs to be stable. Other issues are due to the inherent constraints that a physical system imposes, e.g., actuator saturation and the limits of the work space. High-dimensional continuous state–action space causes yet another set of problems, and particularly affects feasibly and efficiency of robot learning. Model-errors and senor noise can have adverse outcome on the learning process. And the fact that a robot needs to generate its own learning data, i. e., the exploration–exploitation tradeoff and ever changing sampling distributions, creates a large number of problems.

We note that there is no clear receipe for robot learning, but rather a large number of ingredients that, when used appropriately, can achieve excellent learning results. As a caveat, even minor incorrect choices can completely prevent any success of learning. That robot learning has yet not reached maturity is obvious from the fact that learning algorithms are rarely used on robots on a daily basis – most results are feasibility studies and not robust enough for daily use. Many results are also overtuned to a particular physical robot, and do not generalize to others easily.

Avidbots Ex1016
Page 427 of 2154

Some of the open questions for the future of robot learning include:

- How to choose appropriate representations automatically for model learning, value function learning, and policy learning? Perhaps, there is one particular choice that robustly works for many different robots, or maybe one could find a small set of possible approaches that can easily be compared.
- How to create useful reward functions? This topic connects to inverse RL, and on a higher level to how to understand the intentions of observed behavior.
- How much can prior knowledge help? How much is needed? How should it be provided?
- How to integrate more tightly with perception? Robot learning is largely *action-centric* and assumes that *perception* is provided. In reality, there is a perception-action-learning loop where the different components interact significantly, and need to be jointly developed.
- How to reduce parameter sensitivity? Manual tuning of hyperparameters such as gradient rates, forgetting rates, exploration rates, etc., are a common curse for the robot learning practitioner, and often, a small change in a parameter determines success or failure.
- How to robustly deal with modeling errors? Models are great when they work, but disastrous if they are not accurate enough. Probabilistic and robust control methods may help, but can also degrade performance due to very conservative learning results.

This list is by no means exhaustive, but captures some of the key issues. In the end, there is strong need for researchers and scientists who are willing to tackle a complex mixture of theoretical problems and experimental problems. Often, just setting up an experimental robot environment with accurate and efficient debugging and visualization tools is a formidable effort. Then, finding the right experiment and the right data trace that allows for error diagnosis in robot learning is still a bit of an art and requires fairly deep insights into physics, algorithms, software architectures, and technology.

Video-References

VIDEO 352 Inverted helicopter hovering
available from http://handbookofrobotics.org/view-chapter/15/videodetails/352

VIDEO 353 Inverse reinforcement
available from http://handbookofrobotics.org/view-chapter/15/videodetails/353

VIDEO 354 Machine learning table tennis
available from http://handbookofrobotics.org/view-chapter/15/videodetails/354

VIDEO 355 Learning motor primitives
available from http://handbookofrobotics.org/view-chapter/15/videodetails/355

References

15.1 S. Schaal: The new robotics – Towards human-centered machines, HFSP J. Front. Interdiscip. Res, Life Sci. **1**(2), 115–126 (2007)

15.2 B.D. Ziebart, A. Maas, J.A. Bagnell, A.K. Dey: Maximum entropy inverse reinforcement learning, AAAI Conf. Artif. Intell. (2008)

15.3 S. Thrun, W. Burgard, D. Fox: *Probabilistic Robotics* (MIT, Cambridge 2005)

15.4 B. Apolloni, A. Ghosh, F. Alpaslan, L.C. Jain, S. Patnaik (Eds.): *Machine Learning and Robot Perception*, Stud. Comput. Intell., Vol. 7 (Springer, Berlin, Heidelberg 2005)

15.5 O. Jenkins, R. Bodenheimer, R. Peters: Manipulation manifolds: Explorations into uncovering manifolds in sensory-motor spaces, Int. Conf. Dev. Learn. (2006)

15.6 M. Toussaint: Machine learning and robotics, Tutor. Conf. Mach. Learn. (2011)

15.7 D.P. Bertsekas: *Dynamic Programming and Optimal Control* (Athena Scientific, Nashua 1995)

15.8 R.E. Kalman: When is a linear control system optimal?, J. Basic Eng. **86**(1), 51–60 (1964)

15.9 D. Nguyen-Tuong, J. Peters: Model learning in robotics: A survey, Cogn. Process. **12**(4), 319–340 (2011)

15.10 J. Kober, D. Bagnell, J. Peters: Reinforcement learning in robotics: A survey, Int. J. Robotics Res. **32**(11), 1238–1274 (2013)

15.11 J.H. Connell, S. Mahadevan: *Robot Learning* (Kluwer Academic, Dordrecht 1993)

15.12 J. Ham, Y. Lin, D.D. Lee: Learning nonlinear appearance manifolds for robot localization, Int. Conf. Intell. Robots Syst. (2005)

15.13 R.S. Sutton, A.G. Barto: *Reinforcement Learning* (MIT, Cambridge 1998)

15.14 D. Nguyen-Tuong, J. Peters: Model learning with local Gaussian process regression, Adv. Robotics **23**(15), 2015–2034 (2009)

15.15 J. Nakanishi, R. Cory, M. Mistry, J. Peters, S. Schaal: Operational space control: A theoretical and em-

Avidbots Ex1016
Page 428 of 2154

prical comparison, Int. J. Robotics Res. **27**(6), 737–757 (2008)
15.16 F.R. Reinhart, J.J. Steil: Attractor-based computation with reservoirs for online learning of inverse kinematics, Proc. Eur. Symp. Artif. Neural Netw. (2009)
15.17 J. Ting, M. Kalakrishnan, S. Vijayakumar, S. Schaal: Bayesian kernel shaping for learning control, Adv. Neural Inform. Process. Syst., Vol. 21 (2008) pp. 1673–1680
15.18 J. Steffen, S. Klanke, S. Vijayakumar, H.J. Ritter: Realising dextrous manipulation with structured manifolds using unsupervised kernel regression with structural hints, ICRA 2009 Workshop: Approaches Sens. Learn. Humanoid Robots, Kobe (2009)
15.19 S. Klanke, D. Lebedev, R. Haschke, J.J. Steil, H. Ritter: Dynamic path planning for a 7-dof robot arm, Proc. 2009 IEEE Int. Conf. Intell. Robots Syst. (2006)
15.20 A. Angelova, L. Matthies, D. Helmick, P. Perona: Slip prediction using visual information, Proc. Robotics Sci. Syst., Philadelphia (2006)
15.21 M. Kalakrishnan, J. Buchli, P. Pastor, S. Schaal: Learning locomotion over rough terrain using terrain templates, IEEE Int. Conf. Intell. Robots Syst. (2009)
15.22 N. Hawes, J.L. Wyatt, M. Sridharan, M. Kopicki, S. Hongeng, I. Calvert, A. Sloman, G.-J. Kruijff, H. Jacobsson, M. Brenner, D. Skočaj, A. Vrečko, N. Majer, M. Zillich: The playmate system, Cognit. Syst. **8**, 367–393 (2010)
15.23 D. Skočaj, M. Kristan, A. Vrečko, A. Leonardis, M. Fritz, M. Stark, B. Schiele, S. Hongeng, J.L. Wyatt: Multi-modal learning, Cogn. Syst. **8**, 265–309 (2010)
15.24 O.J. Smith: A controller to overcome dead-time, Instrum. Soc. Am. J. **6**, 28–33 (1959)
15.25 K.S. Narendra, A.M. Annaswamy: *Stable Adaptive Systems* (Prentice Hall, New Jersey 1989)
15.26 S. Nicosia, P. Tomei: Model reference adaptive control algorithms for industrial robots, Automatica **20**, 635–644 (1984)
15.27 J.M. Maciejowski: *Predictive Control with Constraints* (Prentice Hall, New Jersey 2002)
15.28 R.S. Sutton: Dyna, an integrated architecture for learning, planning, and reacting, SIGART Bulletin **2**(4), 160–163 (1991)
15.29 C.G. Atkeson, J. Morimoto: Nonparametric representation of policies and value functions: A trajectory-based approach, Adv. Neural Inform. Process. Syst., Vol. 15 (2002)
15.30 A.Y. Ng, A. Coates, M. Diel, V. Ganapathi, J. Schulte, B. Tse, E. Berger, E. Liang: Autonomous inverted helicopter flight via reinforcement learning, Proc. 11th Int. Symp. Exp. Robotics (2004)
15.31 C.E. Rasmussen, M. Kuss: Gaussian processes in reinforcement learning, Adv. Neural Inform. Process. Syst., Vol. 16 (2003) pp. 751–758
15.32 A. Rottmann, W. Burgard: Adaptive autonomous control using online value iteration with Gaussian processes, Proc. IEEE Int. Conf. Robotics Autom. (2009)
15.33 J.-J.E. Slotine, W. Li: *Applied Nonlinear Control* (Prentice Hall, Upper Saddle River 1991)
15.34 A. De Luca, P. Lucibello: A general algorithm for dynamic feedback linearization of robots with elastic joints, Proc. IEEE Int. Conf. Robotics Autom. (1998)
15.35 I. Jordan, D. Rumelhart: Forward models: Supervised learning with a distal teacher, Cognit. Sci. **16**, 307–354 (1992)
15.36 D.M. Wolpert, M. Kawato: Multiple paired forward and inverse models for motor control, Neural Netw. **11**, 1317–1329 (1998)
15.37 M. Kawato: Internal models for motor control and trajectory planning, Curr. Opin. Neurobiol. **9**(6), 718–727 (1999)
15.38 D.M. Wolpert, R.C. Miall, M. Kawato: Internal models in the cerebellum, Trends Cogn. Sci. **2**(9), 338–347 (1998)
15.39 N. Bhushan, R. Shadmehr: Evidence for a forward dynamics model in human adaptive motor control, Adv. Neural Inform. Process. Syst., Vol. 11 (1999) pp. 3–9
15.40 K. Narendra, J. Balakrishnan, M. Ciliz: Adaptation and learning using multiple models, switching and tuning, IEEE Control Syst, Mag. **15**(3), 37–51 (1995)
15.41 K. Narendra, J. Balakrishnan: Adaptive control using multiple models, IEEE Trans. Autom. Control **42**(2), 171–187 (1997)
15.42 M. Haruno, D.M. Wolpert, M. Kawato: Mosaic model for sensorimotor learning and control, Neural Comput. **13**(10), 2201–2220 (2001)
15.43 J. Peters, S. Schaal: Learning to control in operational space, Int. J. Robotics Res. **27**(2), 197–212 (2008)
15.44 H. Akaike: Autoregressive model fitting for control, Ann. Inst. Stat. Math. **23**, 163–180 (1970)
15.45 R.M.C. De Keyser, A.R.V. Cauwenberghe: A self-tuning multistep predictor application, Automatica **17**, 167–174 (1980)
15.46 S.S. Billings, S. Chen, G. Korenberg: Identification of mimo nonlinear systems using a forward-regression orthogonal estimator, Int. J. Control **49**, 2157–2189 (1989)
15.47 E. Mosca, G. Zappa, J.M. Lemos: Robustness of multipredictor adaptive regulators: MUSMAR, Automatica **25**, 521–529 (1989)
15.48 J. Kocijan, R. Murray-Smith, C. Rasmussen, A. Girard: Gaussian process model based predictive control, Proc. Am. Control Conf. (2004)
15.49 A. Girard, C.E. Rasmussen, J.Q. Candela, R.M. Smith: Gaussian process priors with uncertain inputs application to multiple-step ahead time series forecasting, Adv. Neural Inform. Process. Syst., Vol. 15 (2002) pp. 545–552
15.50 C.G. Atkeson, A. Moore, S. Stefan: Locally weighted learning for control, AI Review **11**, 75–113 (1997)
15.51 L. Ljung: *System Identification – Theory for the User* (Prentice-Hall, New Jersey 2004)

Avidbots Ex1016
Page 429 of 2154

15.52 S. Haykin: *Neural Networks: A Comprehensive Foundation* (Prentice Hall, New Jersey 1999)
15.53 J.J. Steil: Backpropagation-decorrelation: Online recurrent learning with O(N) complexity, Proc. Int. Jt. Conf. Neural Netw. (2004)
15.54 C.E. Rasmussen, C.K. Williams: *Gaussian Processes for Machine Learning* (MIT, Cambridge 2006)
15.55 B. Schölkopf, A. Smola: *Learning with Kernels: Support Vector Machines, Regularization, Optimization and Beyond* (MIT, Cambridge 2002)
15.56 K.J. Aström, B. Wittenmark: *Adaptive Control* (Addison Wesley, Boston 1995)
15.57 F.J. Coito, J.M. Lemos: A long-range adaptive controller for robot manipulators, Int. J. Robotics Res. **10**, 684–707 (1991)
15.58 P. Vempaty, K. Cheok, R. Loh: Model reference adaptive control for actuators of a biped robot locomotion, Proc. World Congr. Eng. Comput. Sci. (2009)
15.59 J.R. Layne, K.M. Passino: Fuzzy model reference learning control, J. Intell. Fuzzy Syst. **4**, 33–47 (1996)
15.60 J. Nakanishi, J.A. Farrell, S. Schaal: Composite adaptive control with locally weighted statistical learning, Neural Netw. **18**(1), 71–90 (2005)
15.61 J.J. Craig: *Introduction to Robotics: Mechanics and Control* (Prentice Hall, Upper Saddle River 2004)
15.62 M.W. Spong, S. Hutchinson, M. Vidyasagar: *Robot Dynamics and Control* (Wiley, New York 2006)
15.63 S. Schaal, C.G. Atkeson, S. Vijayakumar: Scalable techniques from nonparametric statistics for real-time robot learning, Appl. Intell. **17**(1), 49–60 (2002)
15.64 H. Cao, Y. Yin, D. Du, L. Lin, W. Gu, Z. Yang: Neural network inverse dynamic online learning control on physical exoskeleton, 13th Int. Conf. Neural Inform. Process. (2006)
15.65 C.G. Atkeson, C.H. An, J.M. Hollerbach: Estimation of inertial parameters of manipulator loads and links, Int. J. Robotics Res. **5**(3), 101–119 (1986)
15.66 E. Burdet, B. Sprenger, A. Codourey: Experiments in nonlinear adaptive control, Int. Conf. Robotics Autom. **1**, 537–542 (1997)
15.67 E. Burdet, A. Codourey: Evaluation of parametric and nonparametric nonlinear adaptive controllers, Robotica **16**(1), 59–73 (1998)
15.68 K.S. Narendra, A.M. Annaswamy: Persistent excitation in adaptive systems, Int. J. Control **45**, 127–160 (1987)
15.69 H.D. Patino, R. Carelli, B.R. Kuchen: Neural networks for advanced control of robot manipulators, IEEE Trans. Neural Netw. **13**(2), 343–354 (2002)
15.70 D. Nguyen-Tuong, J. Peters: Incremental sparsification for real-time online model learning, Neurocomputing **74**(11), 1859–1867 (2011)
15.71 D. Nguyen-Tuong, J. Peters: Using model knowledge for learning inverse dynamics, Proc. IEEE Int. Conf. Robotics Autom. (2010)
15.72 S.S. Ge, T.H. Lee, E.G. Tan: Adaptive neural network control of flexible joint robots based on feedback linearization, Int. J. Syst. Sci. **29**(6), 623–635 (1998)
15.73 C.M. Chow, A.G. Kuznetsov, D.W. Clarke: Successive one-step-ahead predictions in multiple model predictive control, Int. J. Control **29**, 971–979 (1998)
15.74 M. Kawato: Feedback error learning neural network for supervised motor learning. In: *Advanced Neural Computers*, ed. by R. Eckmiller (Elsevier, North-Holland, Amsterdam 1990) pp. 365–372
15.75 J. Nakanishi, S. Schaal: Feedback error learning and nonlinear adaptive control, Neural Netw. **17**(10), 1453–1465 (2004)
15.76 T. Shibata, C. Schaal: Biomimetic gaze stabilization based on feedback-error learning with nonparametric regression networks, Neural Netw. **14**(2), 201–216 (2001)
15.77 H. Miyamoto, M. Kawato, T. Setoyama, R. Suzuki: Feedback-error-learning neural network for trajectory control of a robotic manipulator, Neural Netw. **1**(3), 251–265 (1988)
15.78 H. Gomi, M. Kawato: Recognition of manipulated objects by motor learning with modular architecture networks, Neural Netw. **6**(4), 485–497 (1993)
15.79 A. D'Souza, S. Vijayakumar, S. Schaal: Learning inverse kinematics, IEEE Int. Conf. Intell. Robots Syst. (2001)
15.80 S. Vijayakumar, S. Schaal: Locally weighted projection regression: An O(N) algorithm for incremental real time learning in high dimensional space, Proc. 16th Int. Conf. Mach. Learn. (2000)
15.81 M. Toussaint, S. Vijayakumar: Learning discontinuities with products-of-sigmoids for switching between local models, Proc. 22nd Int. Conf. Mach. Learn. (2005)
15.82 J. Tenenbaum, V. de Silva, J. Langford: A global geometric framework for nonlinear dimensionality reduction, Science **290**, 2319–2323 (2000)
15.83 S. Roweis, L. Saul: Nonlinear dimensionality reduction by locally linear embedding, Science **290**, 2323 (2000)
15.84 H. Hoffman, S. Schaal, S. Vijayakumar: Local dimensionality reduction for non-parametric regression, Neural Process. Lett. **29**(2), 109–131 (2009)
15.85 S. Thrun, T. Mitchell: Lifelong robot learning, Robotics Auton. Syst. **15**, 25–46 (1995)
15.86 Y. Engel, S. Mannor, R. Meir: Sparse online greedy support vector regression, Eur. Conf. Mach. Learn. (2002)
15.87 A.J. Smola, B. Schölkopf: A tutorial on support vector regression, Stat. Comput. **14**(3), 199–222 (2004)
15.88 C.E. Rasmussen: Evaluation of Gaussian Processes and Other Methods for Non-Linear Regression (University of Toronto, Toronto 1996)
15.89 L. Bottou, O. Chapelle, D. DeCoste, J. Weston: *Large-Scale Kernel Machines* (MIT, Cambridge 2007)
15.90 J.Q. Candela, C.E. Rasmussen: A unifying view of sparse approximate Gaussian process regression, J. Mach. Learn. Res. **6**, 1939–1959 (2005)

Avidbots Ex1016
Page 430 of 2154

15.91 R. Genov, S. Chakrabartty, G. Cauwenberghs: Silicon support vector machine with online learning, Int. J. Pattern Recognit. Articial Intell. **17**, 385–404 (2003)

15.92 S. Vijayakumar, A. D'Souza, S. Schaal: Incremental online learning in high dimensions, Neural Comput. **12**(11), 2602–2634 (2005)

15.93 B. Schölkopf, P. Simard, A. Smola, V. Vapnik: Prior knowledge in support vector kernel, Adv. Neural Inform. Process. Syst., Vol. 10 (1998) pp. 640–646

15.94 E. Krupka, N. Tishby: Incorporating prior knowledge on features into learning, Int. Conf. Artif. Intell. Stat. (San Juan, Puerto Rico 2007)

15.95 A. Smola, T. Friess, B. Schoelkopf: Semiparametric support vector and linear programming machines, Adv. Neural Inform. Process. Syst., Vol. 11 (1999) pp. 585–591

15.96 B.J. Kröse, N. Vlassis, R. Bunschoten, Y. Motomura: A probabilistic model for appearance-based robot localization, Image Vis. Comput. **19**, 381–391 (2001)

15.97 M.K. Titsias, N.D. Lawrence: Bayesian Gaussian process latent variable model, Proc. 13th Int. Conf. Artif. Intell. Stat. (2010)

15.98 R. Jacobs, M. Jordan, S. Nowlan, G.E. Hinton: Adaptive mixtures of local experts, Neural Comput. **3**, 79–87 (1991)

15.99 S. Calinon, F. D'halluin, E. Sauser, D. Caldwell, A. Billard: A probabilistic approach based on dynamical systems to learn and reproduce gestures by imitation, IEEE Robotics Autom. Mag. **17**, 44–54 (2010)

15.100 V. Treps: A bayesian committee machine, Neural Comput. **12**(11), 2719–2741 (2000)

15.101 L. Csato, M. Opper: Sparse online Gaussian processes, Neural Comput. **14**(3), 641–668 (2002)

15.102 D.H. Grollman, O.C. Jenkins: Sparse incremental learning for interactive robot control policy estimation, IEEE Int. Conf. Robotics Autom., Pasadena (2008)

15.103 M. Seeger: Gaussian processes for machine learning, Int. J. Neural Syst. **14**(2), 69–106 (2004)

15.104 C. Plagemann, S. Mischke, S. Prentice, K. Kersting, N. Roy, W. Burgard: Learning predictive terrain models for legged robot locomotion, Proc. IEEE Int. Conf. Intell. Robots Syst. (2008)

15.105 J. Ko, D. Fox: GP-bayesfilters: Bayesian filtering using Gaussian process prediction and observation models, Auton. Robots **27**(1), 75–90 (2009)

15.106 J.P. Ferreira, M. Crisostomo, A.P. Coimbra, B. Ribeiro: Simulation control of a biped robot with support vector regression, IEEE Int. Symp. Intell. Signal Process. (2007)

15.107 R. Pelossof, A. Miller, P. Allen, T. Jebara: An SVM learning approach to robotic grasping, IEEE Int. Conf. Robotics Autom. (2004)

15.108 J. Ma, J. Theiler, S. Perkins: Accurate on-line support vector regression, Neural Comput. **15**, 2683–2703 (2005)

15.109 Y. Choi, S.Y. Cheong, N. Schweighofer: Local online support vector regression for learning control, Proc. IEEE Int. Symp. Comput. Intell. Robotics Autom. (2007)

15.110 J.-A. Ting, A. D'Souza, S. Schaal: Bayesian robot system identification with input and output noise, Neural Netw. **24**(1), 99–108 (2011)

15.111 S. Nowlan, G.E. Hinton: Evaluation of adaptive mixtures of competing experts, Adv. Neural Inform. Process. Syst., Vol. 3 (1991) pp. 774–780

15.112 V. Treps: Mixtures of Gaussian processes, Adv. Neural Inform. Process. Syst., Vol. 13 (2001) pp. 654–660

15.113 C.E. Rasmussen, Z. Ghahramani: Infinite mixtures of Gaussian process experts, Adv. Neural Inform. Process. Syst., Vol. 14 (2002) pp. 881–888

15.114 T. Hastie, R. Tibshirani, J. Friedman: *The Elements of Statistical Learning* (Springer, New York, 2001)

15.115 W.K. Haerdle, M. Mueller, S. Sperlich, A. Werwatz: *Nonparametric and Semiparametric Models* (Springer, New York 2004)

15.116 D.J. MacKay: A practical Bayesian framework for back-propagation networks, Computation **4**(3), 448–472 (1992)

15.117 R.M. Neal: *Bayesian Learning for Neural Networks*, Lecture Notes in Statistics, Vol. 118 (Springer, New York 1996)

15.118 B. Schölkopf, A.J. Smola, R. Williamson, P.L. Bartlett: New support vector algorithms, Neural Comput. **12**(5), 1207–1245 (2000)

15.119 C. Plagemann, K. Kersting, P. Pfaff, W. Burgard: Heteroscedastic Gaussian process regression for modeling range sensors in mobile robotics, Snowbird Learn. Workshop (2007)

15.120 W.S. Cleveland, C.L. Loader: Smoothing by local regression: Principles and methods. In: *Statistical Theory and Computational Aspects of Smoothing*, ed. by W. Härdle, M.G. Schimele (Physica, Heidelberg 1996)

15.121 J. Fan, I. Gijbels: *Local Polynomial Modelling and Its Applications* (Chapman Hall, New York 1996)

15.122 J. Fan, I. Gijbels: Data driven bandwidth selection in local polynomial fitting, J. R. Stat. Soc. **57**(2), 371–394 (1995)

15.123 A. Moore, M.S. Lee: Efficient algorithms for minimizing cross validation error, Proc. 11th Int. Conf. Mach. Learn. (1994)

15.124 A. Moore: Fast, robust adaptive control by learning only forward models, Adv. Neural Inform. Process. Syst., Vol. 4 (1992) pp. 571–578

15.125 C.G. Atkeson, A.W. Moore, S. Schaal: Locally weighted learning for control, Artif. Intell. Rev. **11**, 75–113 (1997)

15.126 G. Tevatia, S. Schaal: Efficient Inverse Kinematics Algorithms for High-Dimensional Movement Systems (University of Southern California, Los Angeles 2008)

15.127 C.G. Atkeson, A.W. Moore, S. Schaal: Locally weighted learning, Artif. Intell. Rev. **11**(1–5), 11–73 (1997)

15.128 N.U. Edakunni, S. Schaal, S. Vijayakumar: Kernel carpentry for online regression using randomly varying coefficient model, Proc. 20th Int. Jt. Conf. Artif. Intell. (2007)

Avidbots Ex1016
Page 431 of 2154

15.129 D.H. Jacobson, D.Q. Mayne: *Differential Dynamic Programming* (American Elsevier, New York 1973)
15.130 C.G. Atkeson, S. Schaal: Robot learning from demonstration, Proc. 14th Int. Conf. Mach. Learn. (1997)
15.131 J. Morimoto, G. Zeglin, C.G. Atkeson: Minimax differential dynamic programming: Application to a biped walking robot, Proc. 2009 IEEE Int. Conf. Intell. Robots Syst. (2003)
15.132 P. Abbeel, A. Coates, M. Quigley, A.Y. Ng: An application of reinforcement learning to aerobatic helicopter flight, Adv. Neural Inform. Process. Syst., Vol. 19 (2007) pp. 1–8
15.133 P.W. Glynn: Likelihood ratio gradient estimation: An overview, Proc. Winter Simul. Conf. (1987)
15.134 A.Y. Ng, M. Jordan: Pegasus: A policy search method for large MDPs and POMDPs, Proc. 16th Conf. Uncertain. Artif. Intell. (2000)
15.135 B.M. Akesson, H.T. Toivonen: A neural network model predictive controller, J. Process Control **16**(9), 937–946 (2006)
15.136 D. Gu, H. Hu: Predictive control for a car-like mobile robot, Robotics Auton. Syst. **39**, 73–86 (2002)
15.137 E.A. Wan, A.A. Bogdanov: Model predictive neural control with applications to a 6 DOF helicopter model, Proc. Am. Control Conf. (2001)
15.138 O. Khatib: A unified approach for motion and force control of robot manipulators: The operational space formulation, J. Robotics Autom. **3**(1), 43–53 (1987)
15.139 J. Peters, M. Mistry, F.E. Udwadia, J. Nakanishi, S. Schaal: A unifying methodology for robot control with redundant dofs, Auton. Robots **24**(1), 1–12 (2008)
15.140 C. Salaun, V. Padois, O. Sigaud: Control of redundant robots using learned models: An operational space control approach, Proc. IEEE Int. Conf. Intell. Robots Syst. (2009)
15.141 F.R. Reinhart, J.J. Steil: Recurrent neural associative learning of forward and inverse kinematics for movement generation of the redundant PA-10 robot, Symp. Learn. Adapt. Behav. Robotics Syst. (2008)
15.142 J.Q. Candela, C.E. Rasmussen, C.K. Williams: *Large Scale Kernel Machines* (MIT, Cambridge 2007)
15.143 S. Ben-David, R. Schuller: Exploiting task relatedness for multiple task learning, Proc. Conf. Learn. Theory (2003)
15.144 I. Tsochantaridis, T. Joachims, T. Hofmann, Y. Altun: Large margin methods for structured and interdependent output variables, J. Mach. Learn. Res. **6**, 1453–1484 (2005)
15.145 O. Chapelle, B. Schölkopf, A. Zien: *Semi-Supervised Learning* (MIT, Cambridge 2006)
15.146 J.D. Lafferty, A. McCallum, F.C.N. Pereira: Conditional random fields: Probabilistic models for segmenting and labeling sequence data, Proc. 18th Int. Conf. Mach. Learn. (2001)
15.147 K. Muelling, J. Kober, O. Kroemer, J. Peters: Learning to select and generalize striking movements in robot table tennis, Int. J. Robotics Res. **32**(3), 263–279 (2012)
15.148 S. Mahadevan, J. Connell: Automatic programming of behavior-based robots using reinforcement learning, Artif. Intell. **55**(2/3), 311–365 (1992)
15.149 V. Gullapalli, J.A. Franklin, H. Benbrahim: Acquiring robot skills via reinforcement learning, IEEE Control Syst. Mag. **14**(1), 13–24 (1994)
15.150 J.A. Bagnell, J.C. Schneider: Autonomous helicopter control using reinforcement learning policy search methods, IEEE Int. Conf. Robotics Autom. (2001)
15.151 S. Schaal: Learning from demonstration, Adv. Neural Inform. Process. Syst., Vol. 9 (1996) pp. 1040–1046
15.152 W. B. Powell: AI, OR and Control Theory: A Rosetta Stone for Stochastic Optimization, Tech. Rep. (Princeton University, Princeton 2012)
15.153 C.G. Atkeson: Nonparametric model-based reinforcement learning, Adv. Neural Inform. Process. Syst., Vol. 10 (1998) pp. 1008–1014
15.154 A. Coates, P. Abbeel, A.Y. Ng: Apprenticeship learning for helicopter control, Communication ACM **52**(7), 97–105 (2009)
15.155 R.S. Sutton, A.G. Barto, R.J. Williams: Reinforcement learning is direct adaptive optimal control, Am. Control Conf. (1991)
15.156 A.D. Laud: Theory and Application of Reward Shaping in Reinforcement Learning (University of Illinois, Urbana-Champaign 2004)
15.157 M.P. Deisenrot, C.E. Rasmussen: PILCO: A model-based and data-efficient approach to policy search, 28th Int. Conf. Mach. Learn. (2011)
15.158 H. Miyamoto, S. Schaal, F. Gandolfo, H. Gomi, Y. Koike, R. Osu, E. Nakano, Y. Wada, M. Kawato: A Kendama learning robot based on bidirectional theory, Neural Netw. **9**(8), 1281–1302 (1996)
15.159 N. Kohl, P. Stone: Policy gradient reinforcement learning for fast quadrupedal locomotion, IEEE Int. Conf. Robotics Autom. (2004)
15.160 R. Tedrake, T.W. Zhang, H.S. Seung: Learning to walk in 20 minutes, Yale Workshop Adapt. Learn. Syst. (2005)
15.161 J. Peters, S. Schaal: Reinforcement learning of motor skills with policy gradients, Neural Netw. **21**(4), 682–697 (2008)
15.162 J. Peters, S. Schaal: Natural actor-critic, Neurocomputing **71**(7–9), 1180–1190 (2008)
15.163 J. Kober, J. Peters: Policy search for motor primitives in robotics, Adv. Neural Inform. Process. Syst., Vol. 21 (2009) pp. 849–856
15.164 M.P. Deisenroth, C.E. Rasmussen, D. Fox: Learning to control a low-cost manipulator using data-efficient reinforcement learning. In: *Robotics: Science and Systems VII*, ed. by H. Durrand-Whyte, N. Roy, P. Abbeel (MIT, Cambridge 2011)
15.165 L.P. Kaelbling, M.L. Littman, A.W. Moore: Reinforcement learning: A survey, J. Artif. Intell. Res. **4**, 237–285 (1996)
15.166 M.E. Lewis, M.L. Puterman: *The Handbook of Markov Decision Processes: Methods and Applications* (Kluwer, Dordrecht 2001) pp. 89–111
15.167 J. Peters, S. Vijayakumar, S. Schaal: *Linear Quadratic Regulation as Benchmark for Policy*

Avidbots Ex1016
Page 432 of 2154

Gradient Methods, Technical Report (University of Southern California, Los Angeles 2004)
15.168 R.E. Bellman: *Dynamic Programming* (Princeton Univ. Press, Princeton 1957)
15.169 R.S. Sutton, D. McAllester, S.P. Singh, Y. Mansour: Policy gradient methods for reinforcement learning with function approximation, Adv. Neural Inform. Process. Syst., Vol. 12 (1999) pp. 1057–1063
15.170 T. Jaakkola, M.I. Jordan, S.P. Singh: Convergence of stochastic iterative dynamic programming algorithms, Adv. Neural Inform. Process. Syst., Vol. 6 (1993) pp. 703–710
15.171 J. Rust: Using randomization to break the curse of dimensionality, Econometrica **65**(3), 487–516 (1997)
15.172 D.E. Kirk: *Optimal Control Theory* (Prentice-Hall, Englewood Cliffs 1970)
15.173 A. Schwartz: A reinforcement learning method for maximizing undiscounted rewards, Int. Conf. Mach. Learn. (1993)
15.174 C.G. Atkeson, S. Schaal: Robot learning from demonstration, Int. Conf. Mach. Learn. (1997)
15.175 J. Peters, K. Muelling, Y. Altun: Relative entropy policy search, Natl. Conf. Artif. Intell. (2010)
15.176 G. Endo, J. Morimoto, T. Matsubara, J. Nakanishi, G. Cheng: Learning CPG-based biped locomotion with a policy gradient method: Application to a humanoid robot, Int. J. Robotics Res. **27**(2), 213–228 (2008)
15.177 F. Guenter, M. Hersch, S. Calinon, A. Billard: Reinforcement learning for imitating constrained reaching movements, Adv. Robotics **21**(13), 1521–1544 (2007)
15.178 J.Z. Kolter, A.Y. Ng: Policy search via the signed derivative, Robotics Sci. Syst. V, Seattle (2009)
15.179 A.Y. Ng, H.J. Kim, M.I. Jordan, S. Sastry: Autonomous helicopter flight via reinforcement learning, Adv. Neural Inform. Process. Syst., Vol. 16 (2004) pp. 799–806
15.180 J.W. Roberts, L. Moret, J. Zhang, R. Tedrake: From motor to interaction learning in robots, Stud. Comput. Intell. **264**, 293–309 (2010)
15.181 R. Tedrake: Stochastic policy gradient reinforcement learning on a simple 3D biped, IEEE/RSJ Int. Conf. Intell. Robots Syst. (2004)
15.182 F. Stulp, E. Theodorou, M. Kalakrishnan, P. Pastor, L. Righetti, S. Schaal: Learning motion primitive goals for robust manipulation, IEEE/RSJ Int. Conf. Intell. Robots Syst. (2011)
15.183 M. Strens, A. Moore: Direct policy search using paired statistical tests, Int. Conf. Mach. Learn. (2001)
15.184 A.Y. Ng, A. Coates, M. Diel, V. Ganapathi, J. Schulte, B. Tse, E. Berger, E. Liang: Autonomous inverted helicopter flight via reinforcement learning, Int. Symp. Exp. Robotics (2004)
15.185 T. Geng, B. Porr, F. Wörgötter: Fast biped walking with a reflexive controller and real-time policy searching, Adv. Neural Inform. Process. Syst., Vol. 18 (2006) pp. 427–434
15.186 N. Mitsunaga, C. Smith, T. Kanda, H. Ishiguro, N. Hagita: Robot behavior adaptation for human-robot interaction based on policy gradient reinforcement learning, IEEE/RSJ Int. Conf. Intell. Robots Syst. (2005)
15.187 M. Sato, Y. Nakamura, S. Ishii: Reinforcement learning for biped locomotion, Int. Conf. Artif. Neural Netw. (2002)
15.188 R.Y. Rubinstein, D.P. Kroese: *The Cross Entropy Method: A Unified Approach to Combinatorial Optimization, Monte-Carlo Simulation* (Springer, New York 2004)
15.189 D.E. Goldberg: *Genetic Algorithms* (Addision Wesley, New York 1989)
15.190 J.T. Betts: *Practical Methods for Optimal Control Using Nonlinear Programming*, Adv. Design Control, Vol. 3 (SIAM, Philadelphia 2001)
15.191 R.J. Williams: Simple statistical gradient-following algorithms for connectionist reinforcement learning, Mach. Learn. **8**, 229–256 (1992)
15.192 P. Dayan, G.E. Hinton: Using expectation-maximization for reinforcement learning, Neural Comput. **9**(2), 271–278 (1997)
15.193 N. Vlassis, M. Toussaint, G. Kontes, S. Piperidis: Learning model-free robot control by a Monte Carlo EM algorithm, Auton. Robots **27**(2), 123–130 (2009)
15.194 J. Kober, E. Oztop, J. Peters: Reinforcement learning to adjust robot movements to new situations, Proc. Robotics Sci. Syst. Conf. (2010)
15.195 E.A. Theodorou, J. Buchli, S. Schaal: Reinforcement learning of motor skills in high dimensions: A path integral approach, IEEE Int. Conf. Robotics Autom. (2010)
15.196 J.A. Bagnell, A.Y. Ng, S. Kakade, J. Schneider: Policy search by dynamic programming, Adv. Neural Inform. Process. Syst., Vol. 16 (2003) pp. 831–838
15.197 T. Kollar, N. Roy: Trajectory optimization using reinforcement learning for map exploration, Int. J. Robotics Res. **27**(2), 175–197 (2008)
15.198 D. Lizotte, T. Wang, M. Bowling, D. Schuurmans: Automatic gait optimization with Gaussian process regression, Int. Jt. Conf. Artif. Intell. (2007)
15.199 S. Kuindersma, R. Grupen, A.G. Barto: Learning dynamic arm motions for postural recovery, IEEE-RAS Int. Conf. Humanoid Robots (2011)
15.200 M. Tesch, J.G. Schneider, H. Choset: Using response surfaces and expected improvement to optimize snake robot gait parameters, IEEE/RSJ Int. Conf. Intell. Robots Syst. (2011)
15.201 S.-J. Yi, B.-T. Zhang, D. Hong, D.D. Lee: Learning full body push recovery control for small humanoid robots, IEEE Proc. Int. Conf. Robotics Autom. (2011)
15.202 J.A. Boyan, A.W. Moore: Generalization in reinforcement learning: Safely approximating the value function, Adv. Neural Inform. Process. Syst., Vol. 7 (1995) pp. 369–376
15.203 S. Kakade, J. Langford: Approximately optimal approximate reinforcement learning, Int. Conf. Mach. Learn. (2002)
15.204 E. Greensmith, P.L. Bartlett, J. Baxter: Variance reduction techniques for gradient estimates in

Avidbots Ex1016
Page 433 of 2154

reinforcement learning, J. Mach. Learn. Res. **5**, 1471–1530 (2004)
15.205 M.T. Rosenstein, A.G. Barto: Reinforcement learning with supervision by a stable controller, Am. Control Conf. (2004)
15.206 J.N. Tsitsiklis, B. Van Roy: An analysis of temporal-difference learning with function approximation, IEEE Trans. Autom. Control **42**(5), 674–690 (1997)
15.207 J.Z. Kolter, A.Y. Ng: Regularization and feature selection in least-squares temporal difference learning, Int. Conf. Mach. Learn. (2009)
15.208 L.C. Baird, H. Klopf: *Reinforcement Learning with High-Dimensional Continuous Actions*, Technical Report WL-TR-93-1147 (Wright-Patterson Air Force Base, Dayton 1993)
15.209 G.D. Konidaris, S. Osentoski, P. Thomas: Value function approximation in reinforcement learning using the Fourier basis, AAAI Conf. Artif. Intell. (2011)
15.210 J. Peters, K. Muelling, J. Kober, D. Nguyen-Tuong, O. Kroemer: Towards motor skill learning for robotics, Int. Symp. Robotics Res. (2010)
15.211 L. Buşoniu, R. Babuška, B. de Schutter, D. Ernst: *Reinforcement Learning and Dynamic Programming Using Function Approximators* (CRC, Boca Raton 2010)
15.212 A.G. Barto, S. Mahadevan: Recent advances in hierarchical reinforcement learning, Discret. Event Dyn. Syst. **13**(4), 341–379 (2003)
15.213 S. Hart, R. Grupen: Learning generalizable control programs, IEEE Trans. Auton. Mental Dev. **3**(3), 216–231 (2011)
15.214 J.G. Schneider: Exploiting model uncertainty estimates for safe dynamic control learning, Adv. Neural Inform. Process. Syst., Vol. 9 (1997) pp. 1047–1053
15.215 J.A. Bagnell: Learning Decisions: Robustness, Uncertainty, and Approximation. Dissertation (Robotics Institute, Carnegie Mellon University, Pittsburgh 2004)
15.216 T.M. Moldovan, P. Abbeel: Safe exploration in markov decision processes, 29th Int. Conf. Mach. Learn. (2012)
15.217 T. Hester, M. Quinlan, P. Stone: RTMBA: A real-time model-based reinforcement learning architecture for robot control, IEEE Int. Conf. Robotics Autom. (2012)
15.218 C.G. Atkeson: Using local trajectory optimizers to speed up global optimization in dynamic programming, Adv. Neural Inform. Process. Syst., Vol. 6 (1994) pp. 663–670
15.219 J. Kober, J. Peters: Policy search for motor primitives in robotics, Mach. Learn. **84**(1/2), 171–203 (2010)
15.220 S. Russell: Learning agents for uncertain environments (extended abstract), Conf. Comput. Learn. Theory (1989)
15.221 P. Abbeel, A.Y. Ng: Apprenticeship learning via inverse reinforcement learning, Int. Conf. Mach. Learn. (2004)
15.222 N.D. Ratliff, J.A. Bagnell, M.A. Zinkevich: Maximum margin planning, Int. Conf. Mach. Learn. (2006)
15.223 R.L. Keeney, H. Raiffa: *Decisions with Multiple Objectives: Preferences and Value Tradeoffs* (Wiley, New York 1976)
15.224 N. Ratliff, D. Bradley, J.A. Bagnell, J. Chestnutt: Boosting structured prediction for imitation learning, Adv. Neural Inform. Process. Syst., Vol. 19 (2006) pp. 1153–1160
15.225 D. Silver, J.A. Bagnell, A. Stentz: High performance outdoor navigation from overhead data using imitation learning. In: *Robotics: Science and Systems*, Vol. IV, ed. by O. Brock, J. Trinkle, F. Ramos (MIT, Cambridge 2008)
15.226 D. Silver, J.A. Bagnell, A. Stentz: Learning from demonstration for autonomous navigation in complex unstructured terrain, Int. J. Robotics Res. **29**(12), 1565–1592 (2010)
15.227 N. Ratliff, J.A. Bagnell, S. Srinivasa: Imitation learning for locomotion and manipulation, IEEE-RAS Int. Conf. Humanoid Robots (2007)
15.228 J.Z. Kolter, P. Abbeel, A.Y. Ng: Hierarchical apprenticeship learning with application to quadruped locomotion, Adv. Neural Inform. Process. Syst., Vol. 20 (2007) pp. 769–776
15.229 J. Sorg, S.P. Singh, R.L. Lewis: Reward design via online gradient ascent, Adv. Neural Inform. Process. Syst., Vol. 23 (2010) pp. 2190–2198
15.230 M. Zucker, J.A. Bagnell: Reinforcement planning: RL for optimal planners, IEEE Proc. Int. Conf. Robotics Autom. (2012)
15.231 H. Benbrahim, J.S. Doleac, J.A. Franklin, O.G. Selfridge: Real-time learning: A ball on a beam, Int. Jt. Conf. Neural Netw. (1992)
15.232 B. Nemec, M. Zorko, L. Zlajpah: Learning of a ball-in-a-cup playing robot, Int. Workshop Robotics, Alpe-Adria-Danube Region (2010)
15.233 M. Tokic, W. Ertel, J. Fessler: The crawler, a class room demonstrator for reinforcement learning, Int. Fla. Artif. Intell. Res. Soc. Conf. (2009)
15.234 H. Kimura, T. Yamashita, S. Kobayashi: Reinforcement learning of walking behavior for a four-legged robot, IEEE Conf. Decis. Control (2001)
15.235 R.A. Willgoss, J. Iqbal: Reinforcement learning of behaviors in mobile robots using noisy infrared sensing, Aust. Conf. Robotics Autom. (1999)
15.236 L. Paletta, G. Fritz, F. Kintzler, J. Irran, G. Dorffner: Perception and developmental learning of affordances in autonomous robots, Lect. Notes Comput. Sci. **4667**, 235–250 (2007)
15.237 C. Kwok, D. Fox: Reinforcement learning for sensing strategies, IEEE/RSJ Int. Conf. Intell. Robots Syst. (2004)
15.238 T. Yasuda, K. Ohkura: A reinforcement learning technique with an adaptive action generator for a multi-robot system, Int. Conf. Simul. Adapt. Behav. (2008)
15.239 J.H. Piater, S. Jodogne, R. Detry, D. Kraft, N. Krüger, O. Kroemer, J. Peters: Learning visual representations for perception-action systems, Int. J. Robotics Res. **30**(3), 294–307 (2011)
15.240 M. Asada, S. Noda, S. Tawaratsumida, K. Hosoda: Purposive behavior acquisition for a real robot by vision-based reinforcement learning, Mach. Learn. **23**(2/3), 279–303 (1996)

Avidbots Ex1016
Page 434 of 2154

15.241 M. Huber, R.A. Grupen: A feedback control structure for on-line learning tasks, Robotics Auton. Syst. **22**(3/4), 303–315 (1997)

15.242 P. Fidelman, P. Stone: Learning ball acquisition on a physical robot, Int. Symp. Robotics Autom. (2004)

15.243 V. Soni, S.P. Singh: Reinforcement learning of hierarchical skills on the Sony AIBO robot, Int. Conf. Dev. Learn. (2006)

15.244 B. Nemec, M. Tamošiunaitė, F. Wörgötter, A. Ude: Task adaptation through exploration and action sequencing, IEEE-RAS Int. Conf. Humanoid Robots (2009)

15.245 M.J. Matarić: Reinforcement learning in the multi-robot domain, Auton. Robots **4**, 73–83 (1997)

15.246 M.J. Matarić: Reward functions for accelerated learning, Int. Conf. Mach. Learn. (ICML) (1994)

15.247 R. Platt, R.A. Grupen, A.H. Fagg: Improving grasp skills using schema structured learning, Int. Conf. Dev. Learn. (2006)

15.248 M. Dorigo, M. Colombetti: *Robot Shaping: Developing Situated Agents Through Learning*, Technical Report (International Computer Science Institute, Berkeley 1993)

15.249 G.D. Konidaris, S. Kuindersma, R. Grupen, A.G. Barto: Autonomous skill acquisition on a mobile manipulator, AAAI Conf. Artif. Intell. (2011)

15.250 G.D. Konidaris, S. Kuindersma, R. Grupen, A.G. Barto: Robot learning from demonstration by constructing skill trees, Int. J. Robotics Res. **31**(3), 360–375 (2012)

15.251 A. Cocora, K. Kersting, C. Plagemann, W. Burgard, L. de Raedt: Learning relational navigation policies, IEEE/RSJ Int. Conf. Intell. Robots Syst. (2006)

15.252 D. Katz, Y. Pyuro, O. Brock: Learning to manipulate articulated objects in unstructured environments using a grounded relational representation. In: *Robotics: Science and Systems*, Vol. IV, ed. by O. Brock, J. Trinkle, F. Ramos (MIT, Cambridge 2008)

15.253 C.H. An, C.G. Atkeson, J.M. Hollerbach: *Model-Based Control of a Robot Manipulator* (MIT, Press, Cambridge 1988)

15.254 C. Gaskett, L. Fletcher, A. Zelinsky: Reinforcement learning for a vision based mobile robot, IEEE/RSJ Int. Conf. Intell. Robots Syst. (2000)

15.255 Y. Duan, B. Cui, H. Yang: Robot navigation based on fuzzy RL algorithm, Int. Symp. Neural Netw. (2008)

15.256 H. Benbrahim, J.A. Franklin: Biped dynamic walking using reinforcement learning, Robotics Auton. Syst. **22**(3/4), 283–302 (1997)

15.257 W.D. Smart, L. Pack Kaelbling: A framework for reinforcement learning on real robots, Natl. Conf. Artif. Intell./Innov. Appl. Artif. Intell. (1989)

15.258 D.C. Bentivegna: *Learning from Observation Using Primitives* (Georgia Institute of Technology, Atlanta 2004)

15.259 A. Rottmann, C. Plagemann, P. Hilgers, W. Burgard: Autonomous blimp control using model-free reinforcement learning in a continuous state and action space, IEEE/RSJ Int. Conf. Intell. Robots Syst. (2007)

15.260 K. Gräve, J. Stückler, S. Behnke: Learning motion skills from expert demonstrations and own experience using Gaussian process regression, Jt. Int. Symp. Robotics (ISR) Ger. Conf. Robotics (ROBOTIK) (2010)

15.261 O. Kroemer, R. Detry, J. Piater, J. Peters: Active learning using mean shift optimization for robot grasping, IEEE/RSJ Int. Conf. Intell. Robots Syst. (2009)

15.262 O. Kroemer, R. Detry, J. Piater, J. Peters: Combining active learning and reactive control for robot grasping, Robotics Auton. Syst. **58**(9), 1105–1116 (2010)

15.263 T. Tamei, T. Shibata: Policy gradient learning of cooperative interaction with a robot using user's biological signals, Int. Conf. Neural Inf. Process. (2009)

15.264 A.J. Ijspeert, J. Nakanishi, S. Schaal: Learning attractor landscapes for learning motor primitives, Adv. Neural Inform. Process. Syst., Vol. 15 (2003) pp. 1547–1554

15.265 S. Schaal, P. Mohajerian, A.J. Ijspeert: Dynamics systems vs. optimal control – A unifying view, Prog. Brain Res. **165**(1), 425–445 (2007)

15.266 H.-I. Lin, C.-C. Lai: Learning collision-free reaching skill from primitives, IEEE/RSJ Int. Conf. Intell. Robots Syst. (2012)

15.267 J. Kober, B. Mohler, J. Peters: Learning perceptual coupling for motor primitives, IEEE/RSJ Int. Conf. Intell. Robots Syst. (2008)

15.268 S. Bitzer, M. Howard, S. Vijayakumar: Using dimensionality reduction to exploit constraints in reinforcement learning, Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (2010)

15.269 J. Buchli, F. Stulp, E. Theodorou, S. Schaal: Learning variable impedance control, Int. J. Robotics Res. **30**(7), 820–833 (2011)

15.270 P. Pastor, M. Kalakrishnan, S. Chitta, E. Theodorou, S. Schaal: Skill learning and task outcome prediction for manipulation, IEEE Int. Conf. Robotics Autom. (2011)

15.271 M. Kalakrishnan, L. Righetti, P. Pastor, S. Schaal: Learning force control policies for compliant manipulation, IEEE/RSJ Int. Conf. Intell. Robots Syst. (2011)

15.272 D.C. Bentivegna, C.G. Atkeson, G. Cheng: Learning from observation and practice using behavioral primitives: Marble maze, 11th Int. Symp. Robotics Res. (2004)

15.273 F. Kirchner: Q-learning of complex behaviours on a six-legged walking machine, EUROMICRO Workshop Adv. Mobile Robots (1997)

15.274 J. Morimoto, K. Doya: Acquisition of stand-up behavior by a real robot using hierarchical reinforcement learning, Robotics Auton. Syst. **36**(1), 37–51 (2001)

15.275 J.-Y. Donnart, J.-A. Meyer: Learning reactive and planning rules in a motivationally autonomous animat, Syst. Man Cybern. B **26**(3), 381–395 (1996)

Avidbots Ex1016
Page 435 of 2154

15.276 C. Daniel, G. Neumann, J. Peters: Learning concurrent motor skills in versatile solution spaces, IEEE/RSJ Int. Conf. Intell. Robots Syst. (2012)
15.277 E.C. Whitman, C.G. Atkeson: Control of instantaneously coupled systems applied to humanoid walking, IEEE-RAS Int. Conf. Humanoid Robots (2010)
15.278 X. Huang, J. Weng: Novelty and reinforcement learning in the value system of developmental robots, 2nd Int. Workshop Epigenetic Robotics Model. Cognit. Dev. Robotic Syst. (2002)
15.279 M. Pendrith: Reinforcement learning in situated agents: Some theoretical problems and practical solutions, Eur. Workshop Learn. Robots (1999)
15.280 B. Wang, J.W. Li, H. Liu: A heuristic reinforcement learning for robot approaching objects, IEEE Conf. Robotics Autom. Mechatron. (2006)
15.281 L.P. Kaelbling: Learning in Embedded Systems (Stanford University, Stanford 1990)
15.282 R.S. Sutton: Integrated architectures for learning, planning, and reacting based on approximating dynamic programming, Int. Conf. Mach. Learn. (1990)
15.283 A.W. Moore, C.G. Atkeson: Prioritized sweeping: Reinforcement learning with less data and less time, Mach. Learn. **13**(1), 103–130 (1993)
15.284 J. Peng, R.J. Williams: Incremental multi-step Q-learning, Mach. Learn. **22**(1), 283–290 (1996)
15.285 N. Jakobi, P. Husbands, I. Harvey: Noise and the reality gap: The use of simulation in evolutionary robotics, 3rd Eur. Conf. Artif. Life (1995)

Avidbots Ex1016
Page 436 of 2154

Multimedia Contents

Part B Design

Ed. by Frank C. Park

16 Design and Performance Evaluation
Jorge Angeles, Montreal, Canada
Frank C. Park, Seoul, Korea

17 Limbed Systems
Shuuji Kajita, Tsukuba, Japan
Christian Ott, Wessling, Germany

18 Parallel Mechanisms
Jean-Pierre Merlet, Sophia-Antipolis, France
Clément Gosselin, Quebec, Canada
Tian Huang, Tianjin, China

19 Robot Hands
Claudio Melchiorri, Bologna, Italy
Makoto Kaneko, Suita, Japan

20 Snake-Like and Continuum Robots
Ian D. Walker, Clemson, USA
Howie Choset, Pittsburgh, USA
Gregory S. Chirikjian, Baltimore, USA

21 Actuators for Soft Robotics
Alin Albu-Schäffer, Wessling, Germany
Antonio Bicchi, Pisa, Italy

22 Modular Robots
I-Ming Chen, Singapore, Singapore
Mark Yim, Philadelphia, USA

23 Biomimetic Robots
Kyu-Jin Cho, Seoul, Korea
Robert Wood, Cambridge, USA

24 Wheeled Robots
Woojin Chung, Seoul, Korea
Karl Iagnemma, Cambridge, USA

25 Underwater Robots
Hyun-Taek Choi, Daejeon, Korea
Junku Yuh, Seoul, Korea

26 Flying Robots
Stefan Leutenegger, London, UK
Christoph Hürzeler, Baden, Switzerland
Amanda K. Stowers, Stanford, USA
Kostas Alexis, Zurich, Switzerland
Markus W. Achtelik, Zurich, Switzerland
David Lentink, Stanford, USA
Paul Y. Oh, Las Vegas, USA
Roland Siegwart, Zurich, Switzerland

27 Micro-/Nanorobots
Bradley J. Nelson, Zurich, Switzerland
Lixin Dong, East Lansing, USA
Fumihito Arai, Nagoya, Japan

The chapters contained in Part B, Design, are concerned with the design and modeling of the actual physical realizations of a robot. Some of the more obvious mechanical structures that come to mind are arms, legs, and hands. To this list we can add wheeled vehicles and platforms; snake-like and continuum robots; robots capable of swimming and flying; and robot structures at the micro- and nanoscales. Even for that most basic robotic device, the arm, an incredibly diverse set of structures is possible, depending on the number and types of joints and actuators, and the presence of closed loops in the kinematic structure, or flexibility in the joints and links. Constructing models, and planning and control algorithms for these diverse structures represents an even greater set of challenges.

The topics addressed in these chapters are essential to creating not only the physical robot itself, but also to creating and controlling movements, and manipulating objects in desired ways. As such the connections with the chapters on Robot Foundations (Part A) – particularly the chapters on Kinematics (Chap. 1), Dynamics (Chap. 2), and Mechanisms and Actuation (Chap. 3) – are self-evident. What ultimately distinguishes robotics from other disciplines that study intelligence is that, by definition, robots require a physical manifestation, and by extension must physically interact with the environment. In this regard the topics addressed in these chapters can be said to constitute the most basic layer of this endeavor.

Just as it is difficult to examine human intelligence from a purely abstract perspective, remotely detached from the physical body, so it is difficult to separate the contents of the remaining parts without including in the discussion the actual medium of interaction with the physical world, the (physical) robots themselves. For example, the question of how to coordinate sensing and perception with action (Part C), how to grasp and manipulate objects (Part D), and how to teach robots to move in the world (Part E), must inevitably consider the physical structure of the robot. Robots specialized to various applications and environments (Part F), particularly those intended for direct interaction with humans (Part G), naturally must also consider the robot's physical structure.

With this overview of Part B, we now provide a brief synopsis of each chapter.

Chapter 16, Performance Evaluation and Design Criteria, provides a concise overview of the robot design process, and surveys some of the criteria and tools used in the mechanical design and performance evaluation of robots. Criteria such as workspace volume, local and global dexterity, and elastostatic and elastodynamic performance are not only applicable to determining the topological structure and physical dimensions of the robot, but can also be useful for, e.g., workpiece placement and kinematic redundancy resolution.

Chapter 17, Limbed Systems, discusses the myriad issues involved in the design, analysis, and control of robots with limbs. Defining a limbed system as a robot consisting of a body and at least one limb such that it is able to support and propel itself, the chapter begins with an overview of the design process for limbed systems,from conceptual to detailed, and the basic dynamics of passive and controlled walking. The chapter also examines numerous case studies illustrating the diversity of limbed robot designs, and schemes for their actuation and locomotion. Multi-legged robots, such as dynamic quadrupeds inspired by mammals and behavior-based multilegged robots, are also discussed, as are hybrid leg-wheel-arm robots, tethered walking robots, and even legged robots capable of climbing walls.

Chapter 18, Parallel Mechanisms and Robots, presents an introduction to the kinematics and dynamics of parallel mechanisms such as the well-known Stewart–Gough platform. Parallel mechanisms contain closed loops in their kinematic structure, and as such methods for their analysis differ considerably from those for their serial counterparts. This chapter discusses topics ranging from type synthesis and forward and inverse kinematic solutions of parallel mechanisms, to an investigation of their singularity behavior, workspace characterization, static and dynamic analysis, and practical issues in their design.

Chapter 19, Robot Hands, investigates the principal issues behind the design, modeling, and control of robot hands. Beginning with a discussion of levels of anthropomorphism, and the characterization of robot hand dexterity, the chapter investigates the relevant design issues for robot hands, actuation and transmission architectures, and available sensing technologies. The dynamic modeling and control of robot hands are made challenging not only by the complex kinematic structure, but also by the flexible transmission elements, and the chapter devotes particular attention to these issues.

Chapter 20, Snake-Like and Continuum Robots, begins with a history of snake robots, starting with the pioneering work of Shigeo Hirosein in the early 1970s. While snake-like and continuum robots have very similar exterior appearances, there is considerable diversity in their mechanical design, and the ways in which they are actuated.This chapter describes the mechanical design, actuation, modeling,motion planning, and control of such robots. The chapter also provides case studies of a wide range of existing snake-like and continuum

robots that illustrate the diversity of designs as well as applications.

Chapter 21, Soft Robots, begins with the premise that robots of the future will not resemble the bulky rigid machines found in today's factory floors, but will be compliant and adaptable, and able to safely interact with humans – in other words, soft. This chapter discusses the design, modeling, and control of actuators for this new generation of soft robots.The chapter surveys the different principles and technologies that can be used to design and implement actuators for soft robotics. Many of the concepts are organized so as to allow a direct analogy with natural muscles. Variable impedance actuators are examined in some detail, from their mathematical modeling to motion and force planning and control.

Chapter 22, Modular Robots, provides an overview of the design of modular robots. The chapter begins with a discussion of the concept of modularity, and a definition and classification of modular robots. The chapter then examines reconfigurable modular manipulators, from the earliest designs that were conceived in an industrial automation setting, to more recent self-reconfigurable modular robots. Issues related to the design of modules and the interfaces between them, and the determination of optimal configurations, are also discussed in this chapter.

Chapter 23, Biomimetic Robots, broadly examines the ways in which biological principles can be applied to the design of robotic mechanisms. The challenges of biomimetic design include developing a deep understanding of the relevant natural system, and translating this understanding into a set of engineering design rules; this often entails the development of novel fabrication and actuation to realize the biomimetic design. This chapter discusses the basic design principles underlying biomimetic robots and their contrast with bio-inspired robots, and the fundamental components for developing a biomimetic robot. The chapter also provides detailed reviews of biomimetic designs that have been developed for flapping-wing flight, jumping, crawling, wall climbing, and swimming, as well as the enabling material and fabrication technologies for these biomimetic designs.

Chapter 24, Wheeled Robots, provides a general and comprehensive description of wheeled mobile robots. The chapter begins with a discussion of robot mobility based on the types of wheels and the nature of the kinematic constraints, followed by a classification of wheeled robot structures according to the number and type of wheels and how they are arranged. Omnimobile robots and articulated robot realizations are described, and wheel–terrain interaction models for computing contact forces are also presented. The chapter concludes with a classification of wheel–terrain interaction cases depending on the relative stiffnesses of the wheel and terrain, and the structure and dyamics of suspension systems that enable movement of wheeled robots over uneven surfaces.

Chapter 25, Underwater Robots, examines the design issues for underwater robots, with a focus on remotely operated vehicles and autonomous underwater vehicles. The major components of an underwater robot – from the mechanical elements and subsystems including any attached manipulators, to power sources, actuators and sensors, and architectures for computing and communications and control, are discussed in this chapter. Aspects of the mathematical modeling and control of underwater robots is covered in a separate chapter later in this handbook.

Chapter 26, Flying Robots, provides an overview of the core elements of flying robots. The reader will be guided through the design process of aerial robots, beginning with a qualitative characterization of the different types of flying robot. Design and modeling are particularly closely intertwined in flying robots, and the chapter provides an overview of the underlying aerodynamics and tools for their analysis. The chapter then shows how these tools can be applied to the design and analysis of various types of flying robots, including fixed-wing, rotary-wing, and flapping wing systems, with case studies illustrating these design principles.

Chapter 27, Micro/Nanorobots, provides an overview of the state of the art in micro- and nanorobotics. The former entails robotic manipulation of objects with dimensions in the millimeter to micrometer range, as well as the design and fabrication of autonomous robotic agents within this size range (nanorobotics is defined in the same way, but for dimensions smaller than a micrometer). The chapter outlines scaling effects, actuation, and sensing and fabrication at these scales, and also applications to microassembly, biotechnology, and the construction and characterization of micro- and nano-electromechanical systems.

Multimedia Contents

16. Design and Performance Evaluation

Jorge Angeles, Frank C. Park

In this chapter we survey some of the tools and criteria used in the mechanical design and performance evaluation of robots. Our focus is on robots that are (a) primarily intended for manipulation tasks and (b) constructed with one or more serial kinematic chains. The kinematics of parallel robots is addressed in detail in Chap. 18; their elastostatics is the subject of Sect. 16.5.1. Wheeled robots, walking robots, multifingered hands, and robots intended for outdoor applications, i. e., those encompassing what is known as *field robotics*, are studied in their own chapters; here we provide an overview of the main classes of these robots as relating to design.

16.1 **The Robot Design Process** 400
16.2 **Workspace Criteria** 401
16.2.1 Reaching a Set of Goal Frames 403
16.2.2 Workspace Volume and Topology 404
16.3 **Dexterity Indices** 405
16.3.1 Local Dexterity for Open Chains 405
16.3.2 Dynamics-Based Local Performance Evaluation 407
16.3.3 Global Dexterity Measures 407
16.3.4 Closed-Chain Dexterity Indices ... 407
16.3.5 Alternative Dexterity-Like Measures 407
16.4 **Other Performance Indices** 408
16.4.1 Acceleration Radius 408
16.4.2 Elastostatic Performance 408
16.4.3 Elastodynamic Performance 410
16.5 **Other Robot Types** 411
16.5.1 Robots Mounted on a Fixed Base: Parallel-Kinematics Machines..... 412
16.5.2 Mobile Robots 414
16.5.3 Aquatic Robots 415
16.5.4 Flying Robots 416
16.6 **Summary** 416
References 416

The most obvious application of the criteria and tools described in this chapter is in the mechanical design of a robot. Robot design differs from the design of 1-degree-of-freedom (DOF) machinery in that the latter is intended for one specific task, e.g., picking up a workpiece from a belt conveyor and placing it on a magazine. Moreover, the conveyor is synchronized with the manipulating machine and the magazine is stationary, with well-defined locations where each workpiece is to be placed. Manipulation robots, in contrast, are not intended for one specific task, but rather for a *family of tasks* falling within one class of workpiece motions, e.g., planar, spherical, translational, or motions produced by systems of the selective compliance assembly robot arm (SCARA) type, also known as *Schönflies displacements* [16.1]. The challenge that robot designers face is therefore one of *uncertainty* in the specific task that the robot will be required to execute. Design criteria have been devised to help the designer cope with uncertainty, as discussed herein.

Avidbots Ex1016
Page 440 of 2154

16.1 The Robot Design Process

Given a family of tasks that constitute the *functional requirements* in the design process, besides more-detailed *design specifications*, the role of the designer consists in producing a robot that will meet all the requirements and specifications. The various stages in the robot design job at hand are intended to:

1. Determine the *topology* of the kinematic chain underlying the mechanical structure. Under this item we consider first the robot type: serial, parallel or hybrid. Then, a decision is to be made on the layout of the various *subchains* in terms of the type of joints, most commonly, revolute and prismatic. Recently, one additional type has been recognized to be equally useful, the Π-joint, coupling two links under relative translation by means of two other links undergoing identical angular displacements, although about different parallel axes. The four links form a parallelogram four-bar linkage [16.2].
2. Determine the *geometric dimensions* of the various links defining the *robotic architecture*, as required to fill a table of *Denavit–Hartenberg (DH) parameters* [16.3] so as to satisfy workspace requirements. Although these parameters are usually understood to include the joint variables, these variables do not affect the *robot architecture*; they determine instead the *robot posture*.
3. Determine the structural dimensioning of the various links and joints, as needed to meet static load requirements, where load includes both forces and moments – wrenches – under either the most demanding or the most likely operation conditions, depending on the design philosophy adopted at the outset.
4. Determine the structural dimensioning of the various links and joints, as needed to meet dynamic load requirements, where loads are inertia effects of links and manipulated object.
5. Determine the elastodynamic dimensioning of the overall mechanical structure, including the actuator dynamics, to avoid a specific spectrum of excitation frequencies under either the most demanding or the most likely operation conditions.
6. Select the actuators and their mechanical transmissions for the operation conditions adopted at the outset to cope with task uncertainty.

The above stages can be performed sequentially, in the order given above: (i) first, the topology is determined based on the family of tasks specified at the outset and the *shape* of the workspace, as discussed in Sect. 16.2.2; (ii) the link geometry is defined based on the workspace requirements, which include the *maximum reach*, and the topology defined in stage 1; (iii) with the link geometry thus defined, the structural dimensioning of links and joints (unless the robot under design is parallel, which does not fall within the scope of this chapter, all joints are actuated) is undertaken, so as to support the static loads assumed at the outset; (iv) with the links and joints dimensioned for static-load conditions, the link centers of mass and link inertia matrices are determined for a preliminary evaluation of the motor torque requirements (this evaluation is preliminary in that it does not consider the dynamic load brought about by the actuators; this load can be significant, even in the case of parallel robots, which can have all their motors fixed to the robot base); (v) with the links assumed rigid, joint stiffness is assumed, based on experience or using data from a similar robot, which then leads to an elastodynamic model whose natural modes and frequencies can be determined at a selected set of robot postures (dynamic behavior of the structure is dependent on robot posture) by means of scientific code such as Matlab or computer-aided engineering (CAE) code such as CATIA, Siemens PLM, Pro/Engineer or ANSYS; and (vi) if the frequency spectrum of the robot structure is acceptable, the designer can continue to motor selection; otherwise, a redimensioning is required, which means returning to stage 3.

Even though a design cycle can be completed as outlined above, the designer must now incorporate into the elastodynamic model the structural and inertial data provided by the motor manufacturer. This requires a return to stage 5 and a new elastodynamic analysis. It is thus apparent that the robot design process has one element in common with engineering design in general: both are iterative and open-ended [16.4]. Remarkably, however, the various items driving each design stage are, to a large extent, independent of each other, e.g., topology and geometry can be determined independently from motor selection. Obviously, all issues interact in the overall design process, but, within certain design specifications, the various items do not contradict each other, as to warrant a multiobjective design approach. That is, the optimum design of serial robots can be accomplished fairly well by means of a sequence of single-objective optimization jobs. Again, the results of the last stage, motor selection, must be integrated into an overall mathematical model to test the overall performance. One reference addressing practical optimization issues in the conceptual design of industrial robots is [16.5].

Only when the physical limits of components have been exhausted may a radical redesign requiring a return to stage 1 be warranted. This is the case with SCARA systems. Current industrial robots of this class bear topologies that are largely of the serial type, but some parallel SCARA systems have now appeared in the market. Indeed, the quest for shorter cycle time, as pertaining to an industry test cycle – described in Sect. 16.2.1 – has prompted the industry to look for alternatives to serial architectures. This is how ABBRobotics developed a hybrid parallel-serial robot, the *FlexPicker*, built upon *Clavel*'s Delta robot [16.6], to which a fourth axis has been added in series with the first three. The latter are laid out in a symmetric, parallel architecture that enables Delta to produce pure translations of its moving platform. Adept Technology's Quattro s650H is reportedly capable of three cycles/s for the same test cycle.

This chapter is organized according to the various stages of the robot design process outlined earlier. Noting that topology selection and geometric dimensioning are tightly coupled in the kinematic design process, we first begin with an examination of workspace criteria: we review methods for determining the topology of the kinematic chain, followed by the geometric dimensions so as to satisfy workspace requirements. We then review in detail the various criteria developed for characterizing a robot's manipulating capability, focusing on quantitative notions of dexterity based on both kinematic and dynamic models. We then examine methods for structural dimensioning of the links and joints so as to meet both static and dynamic load requirements. Finally, we discuss elastodynamic dimensioning, and actuator and gear sizing, taking into account properties such as the natural frequency of the robot, and force and acceleration capability requirements. We end the chapter with an overview of mobile, aquatic, and flying robots.

16.2 Workspace Criteria

The most obvious consideration in designing a robot is that its workspace has a set of *required characteristics*. This is a fundamental problem in classical mechanism design, and raises the obvious question of how a user can specify those characteristics.

Issues to consider here pertain, mostly, to what *Vijaykumar* et al. [16.7] termed the *regional structure* of a manipulator. This applies to manipulators with a *decoupled architecture*, whose last three revolutes have concurrent axes, thereby forming a *spherical wrist*, the point of concurrency being the *wrist center*. The manipulation task of architectures of this kind thus allows for a *decoupling* of the positioning and the orientation subtasks: the regional structure, consisting of the first three joints, is first postured so as to locate the center of its wrist at a specified point $C(x, y, z)$; then, the *local structure*, i. e., the wrist, is postured so as to make the end-effector (EE) attain a specified orientation with respect to a frame fixed to the base, given by a rotation matrix.

Most algorithms reported in the literature to determine the workspace of a given robot refer to the workspace of the regional structure. Here, we should distinguish between the workspace of the kinematic chain, regardless of the physical implementation of the chain, and that of the physical robot. In the former, all revolute joints are capable of unlimited rotations about their axes; in the latter, joint limits are needed, for example, to avoid wire entanglement. In the early stages of robot design, joint limits need not be considered, the workspace thus exhibiting symmetries that are proper of the type of joints of the regional structure. If the first joint is a revolute, the workspace has an axis of symmetry, namely, the axis of this revolute joint; if the first joint is prismatic, the workspace has an extrusion symmetry, with the direction of extrusion given by the direction of motion of this joint. As prismatic joints are infinitely extensive, so is the kinematic workspace of a robot with a prismatic joint. The kinematic workspaces of robots with prismatic joints are usually displayed for a finite portion of this workspace.

In the case of parallel robots, to be studied in full detail in Chap. 14, the regional structure is elusive, in general. The usual practice when displaying the workspace for these robots is to assume a constant orientation of the moving plate, the counterpart of the EE of serial robots [16.8]. A common architecture of parallel robots, which arises quite naturally in the design process, entails identical legs symmetrically placed both on the base platform and on the moving platform. Each leg is, in turn, a serial kinematic chain with one or two active joints, all others being passive. The workspace of this kind of robots also exhibits certain symmetries, but no axial symmetry. The symmetries are dictated by the number of legs and the types of actuated joints.

Coming back to serial robots, the workspace can be defined by an envelope that is essentially of one of two types, either a manifold or a surface that is

smooth *almost everywhere*, i. e., smooth everywhere except for a set of points of *measure zero* in the Lebesgue sense [16.10]. Broadly speaking, a set of measure zero on a surface is a curve, e.g., a meridian on a sphere, or a set of isolated points on a line, e.g., the set of rational numbers on the real line. A paradigm for this second kind of workspace is that of the Puma robot, whose kinematic chain is displayed in Fig. 16.1. In this figure, the regional and the local structures are clearly distinguished, the former being *fully extended*. The workspace of this robot is obtained upon locking all joints but the second, when the robot is in the posture shown in Fig. 16.1. Then, the second joint is fully rotated about its axis, the center C of the wrist then describing a circle of radius R equal to the distance of C from the line L_2, the plane of the circle being normal to this line and lying a distance b_3 from the axis L_1 of the first joint. This distance is known as the *shoulder offset*. Now, with all joints locked again, but this time with the first joint unlocked, the robot is turned as a rigid body about L_1. The result is the toroid of Fig. 16.4. Notice that the solid enclosed by this surface is the result of the Boolean operation $S-C$, where S is the sphere of radius R centered at point O_2 of Fig. 16.1, while C is the infinite cylinder of radius b_3 and axis L_1, which appears as Z_1 in Fig. 16.2. It is noteworthy that, although this workspace can be readily generated by a simple Boolean operation, it cannot possibly be generated by an implicit function of the form $f(x, y, z) = 0$ because the surface is not a manifold.

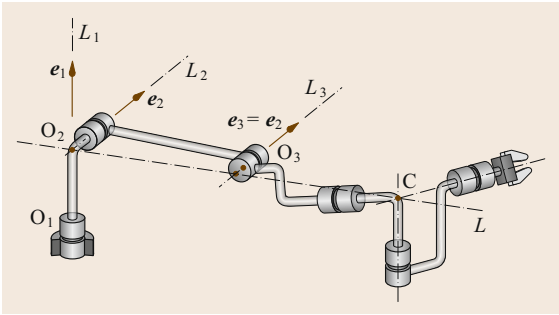


Fig. 16.1 A Puma robot in a fully stretched posture (after [16.9])

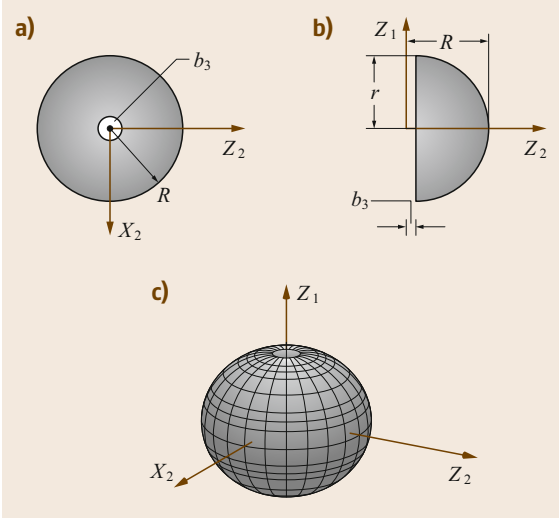


Fig. 16.2 The workspace of a Puma robot (after [16.9])

Robots with manifold workspaces are not common in industry. We display in Fig. 16.3 an architecture for the regional structure of a six-axis decoupled robot, with its neighboring axes mutually orthogonal and at the same distance a from each other. The common normals to the two pairs of axes, X_2 and X_3, also lie at the same distance a, as do X_4 from X_3 and C from Z_3. Point C is the center of the spherical wrist, the latter not being included in the figure. The workspace of this robot admits a representation of the form $f(x, y, z) = 0$ [16.9], which produces the manifold workspace of Fig. 16.4. The shaded internal region of the workspace includes all points admitting four real inverse-kinematics solutions, all other points admitting only two.

Given that any point of the workspace boundary represents a positioning singularity – different from an orientation singularity – manipulators with workspace boundaries that are not manifolds exhibit double singularities at the edges of their workspace boundary, which means that at edge points the rank of the robot Jacobian becomes deficient by two. At any other point of the workspace boundary the rank deficiency is by one.

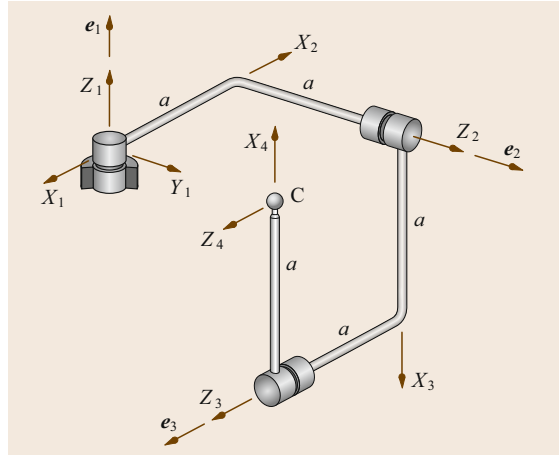


Fig. 16.3 An orthogonal three-revolute robot (after [16.9])

Design rules based on the *shape* of the workspace can now be drawn:

1. If the workspace required is axially symmetric and finite, use a serial robot with a regional structure composed of revolute joints only.
2. If the workspace required has a symmetry of extrusion and is unbounded, use a serial robot with regional structure having one first joint of the prismatic type. Here, *unbounded* is used in a restricted sense, meaning much larger in one direction than the others. Moreover:
   - If one direction is required to be much larger than the others, then practical implementations of prismatic joints are available in the form of rails either overhead, thereby giving rise to *gantry* robots, or on the floor.
   - If two directions are required to be much larger than the other, then use a wheeled mobile robot carrying a manipulator on top. A famous realization of this concept is the National Aeronautical and Space Agency's (NASA) *Sojourner* used in the *Pathfinder* mission to Mars in 1997.
3. If axial symmetry is not required, but rather a workspace with various coplanar axes of symmetry, similar to those of regular polygons, use a parallel robot.

16.2.1 Reaching a Set of Goal Frames

Closely related to the problem of workspace specification is that of task specification. In mechanism design it is customary to specify a set of coordinate frames in space, and to design a mechanism with an a priori specified topology that can visit these frames. An order in which the frames must be reached may be given. In the event that not all of the frames are reachable, then one may seek a mechanism that comes *closest*, in some suitable sense, to the specified frames. The literature on this classical mechanism design problem is vast [16.1, 11, 12] and the references cited therein. Some further remarks in connection with this goal-frame approach to robot dimensioning are noteworthy:

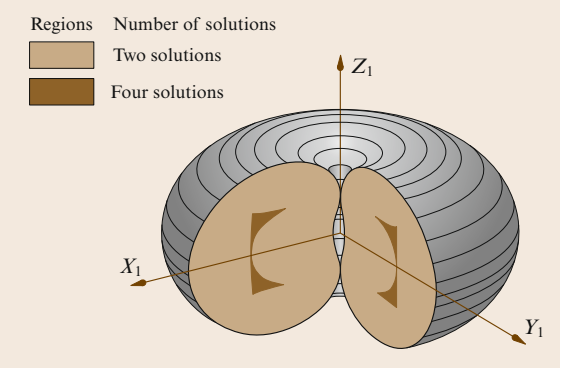


Fig. 16.4 The workspace of the orthogonal robot of Fig. 16.3 (after [16.9])

1. Reaching *exactly* the desired frames may not always be desired or possible: in some cases it is better to use an optimization approach that allows for solutions that will visit the desired poses within a minimum error (provided that an error norm can be suitably engineered, of course, as suggested in [16.13]).
2. It has been claimed [16.8] that interval analysis allows not only a discrete set of desired poses but also a full six-dimensional (6-D) workspace to be met while taking into account manufacturing errors.
3. The branching problem occurring in 1-DOF mechanisms may also occur in robot design: a design solution based on via points may indeed visit the prescribed poses, but not all of these may be reachable within the same assembly mode. This problem is exacerbated in the design of serial robots, as a 6-DOF, revolute-coupled robot may admit up to 16 distinct postures – branches – for one given EE pose [16.14, 15].
4. While a robot designed to visit a set of prescribed poses via its EE will be able to visit that set, we should not forget that the purpose of using robots is first and foremost to be able to perform not one single task, but rather a family of tasks. In this light, the set of poses for which a robot is designed might as well be a task that is *representative* of that family.

In connection with remark 4 above, we can cite the design or evaluation of SCARA systems. A SCARA system is a 4-DOF serial robot capable of tasks that lie within the *Schönflies* subgroup of the group of rigid-body displacements [16.16, 17], namely the set of three-dimensional (3-D) displacements augmented with a rotation about an axis of fixed direction. In these systems, the task at hand is given by two vertical segments joined by one horizontal segment. Moreover, the length of the vertical segments is 25.0 mm, that of the horizontal segment being 300.0 mm. While the EE is traversing the horizontal segment, moreover, it should rotate about a vertical axis through an angle of 180°. This task specification, which has been adopted by SCARA manufacturers, does not indicate how to negotiate the corners, which is left to the imagination of the robotics engineer.

Avidbots Ex1016
Page 444 of 2154

16.2.2 Workspace Volume and Topology

Reachable and Dexterous Workspace

Beginning with the early work of *Roth* [16.18], there have been many studies on the relationship between manipulator kinematic geometry and its workspace. Most studies have focused on a classification of the workspace into two components, the *reachable* and the *dexterous* workspace [16.19]. Given a reference point P attached to a manipulator EE, such as the center of the spherical wrist, or a point on the EE, the reachable workspace is defined to be the set of points in physical space that can be reached by P. The dexterous workspace, on the other hand, is the set of points that can be reached by P with arbitrary EE orientations.

The early literature on workspace focuses on numerical and algebraic methods to characterize these workspaces. Reachable and dexterous workspaces have been analyzed using numerical techniques by *Kumar* and *Waldron* [16.19], *Yang* and *Lee* [16.20], and *Tsai* and *Soni* [16.21], among others. The advantage of these schemes over algebraic approaches is that kinematic constraints can be readily included. More-general design principles or insights, however, are more difficult to come by using these techniques. Among the algebraic approaches to workspace characterization, a topological analysis of robot workspace is given by *Gupta* and *Roth* [16.22] and *Gupta* [16.23]; here the concept of workspace holes and voids is defined, and conditions for their existence are identified. The shape of the reachable and dexterous workspaces is also analyzed as a function of P.

Further studies of workspace analysis were reported by *Freudenstein* and *Primrose* [16.24] and by *Lin* and *Freudenstein* [16.25], where precise relationships between kinematic and workspace parameters are developed, and a class of three-joint manipulators is optimized for workspace volume. A more general analysis of workspace optimization is given in *Vijaykumar* et al. [16.7]. Performance criteria for manipulators are defined here in terms of the dexterous workspace; given that a manipulator satisfies certain constraints on its Denavit–Hartenberg parameters, it is shown that the optimal six-revolute (6R) design is the elbow manipulator.

A typical design of robot regional architecture is of the *orthogonal type*, consisting of one revolute of the vertical axis and two revolutes of the horizontal axes, one of which intersects the vertical axis. Moreover, the most common architecture includes intermediate and distal links of identical lengths. The workspace of this architecture is thus a sphere of radius equal to twice that common link length. The volume of this workspace is thus determined by the length in question. As shown by *Yoshikawa* [16.26], the workspace of the two-link planar manipulator defined by the last two links of the foregoing regional architecture is of maximum area for a prescribed reach and equal link lengths. As a result, the volume of the same regional architecture is similarly of maximum volume for a prescribed reach.

Differential-Geometric Workspace Characterization

Workspace can also be approached from a differential-geometric perspective, by regarding the configuration space of a robotic EE frame as a subset of the special Euclidean group SE(3). An important physical consideration in defining the workspace volume of spatial mechanisms is that it should not depend on the choice of fixed reference frame. Less obvious but just as important is the requirement that the volume should not depend on which point of the last link the EE frame is fixed to. This last condition has the following physical significance: if the EE were enlarged or shrunk, then the robot would have the same workspace volume. The workspace volume of a robot therefore depends only on the joint axes.

The workspace volume of a robot is defined by regarding SE(3) as a Riemannian manifold, so that the workspace volume is simply the volume of the image of the forward kinematic map f with respect to the volume form on SE(3). It is known that SE(3) has a bi-invariant volume form, that is, the notion of volume is invariant with respect to the choice of both the fixed (base) and moving (EE) frames – see, e.g., *Loncaric* [16.27]. *Paden* and *Sastry* [16.28] provide the following visualization for this volume form: Suppose an airplane is restricted to move within a cube of airspace of length 1 km on a side. At each point within this cube the airplane can point itself anywhere in a 4π solid angle and roll 2π about the direction it is pointing. The orientation volume at such an airplane is $4\pi \times 2\pi = 8\pi^2\,\mathrm{rad}^3$. Multiplying by the positional volume one obtains $8\pi^2\,\mathrm{rad}^3\mathrm{km}^3$ for the volume of the free configuration space of the aircraft.

This depiction is the notion of workspace volume used for robots; it has the advantage of being able to trade off orientation freedom for positional freedom smoothly, unlike the popular notion of dexterous workspace. Note that the actual numerical value one obtains will depend on the choice of length scale for physical space; this in itself does not pose a serious problem for workspace volumes, as long as the same length scale is maintained when comparing different workspaces.

In [16.28] *Paden* and *Sastry* show that the optimal 6R manipulator that maximizes the workspace volume subject to a kinematic length constraint is the

elbow manipulator. This result is consistent with the earlier finding of *Vijaykumar* et al. [16.7], but the authors employ the geometric framework outlined above. Moreover, the results are obtained without some of the a priori assumptions on the kinematic structure made by Vijaykumar.

16.3 Dexterity Indices

16.3.1 Local Dexterity for Open Chains

Dexterity can be defined as the ability to move and apply forces and torques in arbitrary directions with equal ease, the concept thus belonging to the realm of *kinetostatics*, which is the study of the interplay between *feasible twists* and *constraint wrenches* in multibody mechanical systems under *static conservative conditions*. Here, twist is the 6-D array of velocity variables of a rigid body, involving three components of a landmark-point velocity and three of angular velocity; wrench, in turn, is the 6-D array of static variables acting on a rigid body, three accounting for the resultant force applied at the same landmark point and three for the concomitant moment acting on the same body.

Salisbury and *Craig* [16.29] introduced the concept of dexterity when working on the design of articulated hands. At issue is the way in which input joint velocity errors propagate to the output velocities of each fingertip. To illustrate this concept, let $\mathbf{J}(\boldsymbol{\theta})$ denote the Jacobian of the forward kinematic map, i. e.,

$$\boldsymbol{t} = \mathbf{J}(\boldsymbol{\theta})\dot{\boldsymbol{\theta}} \,, \tag{16.1}$$

in which $\boldsymbol{\theta}$ and $\dot{\boldsymbol{\theta}}$ denote the vectors of joint variables and joint rates, respectively, while $\boldsymbol{t}$ is the EE *twist*, defined, in turn, as,

$$\boldsymbol{t} = \begin{pmatrix} \boldsymbol{\omega} \\ \dot{\boldsymbol{p}} \end{pmatrix} \,, \tag{16.2}$$

with $\boldsymbol{\omega}$ denoting the angular velocity of the EE and $\dot{\boldsymbol{p}}$ the velocity of the *operation point* P of the EE, at which the task is specified.

Assuming the Jacobian $\mathbf{J}$ is $m \times n$, the singular value decomposition of J can be expressed in the form

$$\mathbf{J} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^{\mathrm{T}} \,, \tag{16.3}$$

where $\mathbf{U}$ and $\mathbf{V}$ are respectively $m \times m$ and $n \times n$ unitary matrices, and $\boldsymbol{\Sigma}$ is a $m \times n$ matrix with zeros everywhere, except for its (i, i) entries, for $i = 1, \ldots n$ if $m > n$; otherwise, for $i = 1, \ldots m$. The non-zero elements are the (non-negative) singular values of $\mathbf{J}$. At nonsingular postures, the Jacobian is invertible and we can write

$$\dot{\boldsymbol{\theta}} = \mathbf{V}\boldsymbol{\Sigma}^{-1}\mathbf{U}^{\mathrm{T}}\boldsymbol{t} \,. \tag{16.4}$$

Furthermore, if we assume that all the components of both $\boldsymbol{t}$ and $\dot{\boldsymbol{\theta}}$ have the same physical units, which is the case for purely positioning or purely orienting manipulators with only revolute joints, then we can take the Euclidean norm of both sides of (16.4), thereby obtaining

$$\|\dot{\boldsymbol{\theta}}\|^2 = \boldsymbol{t}^{\mathrm{T}}\mathbf{U}\left(\boldsymbol{\Sigma}\boldsymbol{\Sigma}^{\mathrm{T}}\right)^{-1}\mathbf{U}^{\mathrm{T}}\boldsymbol{t} \,. \tag{16.5}$$

If we let $\boldsymbol{v} = \mathbf{U}^{\mathrm{T}}\boldsymbol{t}$ then the above expression for $\|\dot{\boldsymbol{\theta}}\|^2$ becomes

$$\boldsymbol{v}^{\mathrm{T}}\left(\boldsymbol{\Sigma}\boldsymbol{\Sigma}^{\mathrm{T}}\right)^{-1}\boldsymbol{v} = \|\dot{\boldsymbol{\theta}}\|^2 \,. \tag{16.6}$$

Now, if the i-th component of $\boldsymbol{v}$ is denoted by v_i, for $i = 1, \ldots, n$, and we look at the mapping of the unit ball in $\mathcal{J}$, $\|\dot{\boldsymbol{\theta}}\|^2 = 1$, (16.6) leads to

$$\frac{v_1}{\sigma_1^2} + \frac{v_2}{\sigma_2^2} + \cdots + \frac{v_n}{\sigma_n^2} = 1 \,, \tag{16.7}$$

which is the canonical equation of an ellipsoid of semiaxes $\{\sigma_i\}_1^n$ in the $\mathcal{G}$-space, i. e., the space of Cartesian velocities, or twists of the EE. Notice that the ellipsoid in question takes its canonical form when represented in a coordinate frame of axes oriented in the directions of the eigenvectors of $\mathbf{U}$.

In summary the unit ball in joint space is mapped by the Jacobian-inverse $\mathbf{J}^{-1}$ into an ellipsoid whose semiaxes are the singular values of $\mathbf{J}$. That is, $\mathbf{J}$ *distorts* the unit ball in the joint-rate space into an ellipsoid in the EE-twist space. Hence, a measure of the quality of motion and force transmission of the robotic architecture from the joints to the EE is given by the above distortion; *the smaller the distortion, the higher the quality of the transmission*.

A measure of the Jacobian-induced distortion can thus be defined as the ratio of the largest σ_{M} to the smallest σ_{m} singular values of $\mathbf{J}$, which is nothing but the *condition number* κ_2 of $\mathbf{J}$ based on the matrix 2-norm [16.30], i. e.,

$$\kappa_2 = \frac{\sigma_{\mathrm{M}}}{\sigma_{\mathrm{m}}} \,. \tag{16.8}$$

Actually, (16.8) is only one possibility of computing the condition number of $\mathbf{J}$, or of any $m \times n$ matrix

for that matter, and certainly not the most economical. Notice that this definition requires knowledge of the singular values of the Jacobian. However, computing the singular values of a matrix is as computationally intensive as computing eigenvalues, with the added cost of a polar decomposition [16.31]; the combined operation is slightly less expensive than the singular-value decomposition [16.32]. The most general definition of condition number, for $n \times n$ matrices, is [16.30]

$$\kappa(\mathbf{A}) = \|\mathbf{A}\|\|\mathbf{A}^{-1}\| \,. \tag{16.9}$$

The expression (16.8) is obtained when the matrix 2-norm is adopted in (16.9). The matrix 2-norm is defined as

$$\|\mathbf{A}\|_2 \equiv \max_i \{\, \sigma_i \,\} \,. \tag{16.10}$$

If, on the other hand, the *weighted matrix Frobenius norm* is adopted, which is defined as

$$\|\mathbf{A}\|_\mathrm{F} \equiv \sqrt{\frac{1}{n}\mathrm{tr}(\mathbf{A}\mathbf{A}^\mathrm{T})} \equiv \sqrt{\frac{1}{n}\mathrm{tr}(\mathbf{A}^\mathrm{T}\mathbf{A})} \,, \tag{16.11}$$

then, apparently, the computation of the singular values can be obviated. When the weight $1/n$ is omitted in the above definition, the standard Frobenius norm is obtained. Notice, however, that the weighted Frobenius norm is more significant in engineering, for it does not depend on the number of rows and columns of the matrix at hand. The weighted Frobenius norm, in fact, yields the root-mean-square (rms) value of the set of singular values.

The *Frobenius condition number* κ_F of the Jacobian $\mathbf{J}$, based on the matrix Frobenius norm, is then

$$\begin{aligned} \kappa_\mathrm{F}(\mathbf{J}) &= \frac{1}{n}\sqrt{\mathrm{tr}(\mathbf{J}\mathbf{J}^\mathrm{T})}\sqrt{\mathrm{tr}[(\mathbf{J}\mathbf{J}^\mathrm{T})^{-1}]} \\ &= \frac{1}{n}\sqrt{\mathrm{tr}(\mathbf{J}^\mathrm{T}\mathbf{J})}\sqrt{\mathrm{tr}[(\mathbf{J}^\mathrm{T}\mathbf{J})^{-1}]} \,. \end{aligned} \tag{16.12}$$

One more important difference between the two forms of computing the matrix condition number is worth pointing out: $\kappa_2(\cdot)$ is not an *analytic function* of its matrix argument, while $\kappa_\mathrm{F}(\cdot)$ is. Hence, the condition number based on the Frobenius norm can be applied to great advantage in robot architecture design, as $\kappa_\mathrm{F}(\cdot)$ is differentiable and lends itself to gradient-dependent optimization methods, which are much faster than direct methods based only on function evaluations. In robot control, which requires real-time computations, $\kappa_\mathrm{F}(\cdot)$ also offers practical advantages, for its computation obviates the knowledge of the singular values, only requiring a matrix inversion, which is a rather simple task.

The significance of the condition number in robot design and control can be best understood if we recall that the concept stems from numerical analysis in connection with the solution of the linear system of equations (16.1) for $\dot{\boldsymbol{\theta}}$. Given that $\mathbf{J}$ is a function of both the architecture parameters and the posture variables $\boldsymbol{\theta}$, $\mathbf{J}$ is known only to within the error with which those quantities are known. Further, let the architecture parameters, namely, the constant values in the Denavit–Hartenberg parameter list, be stored in an array $\boldsymbol{p}$. Both $\boldsymbol{p}$ and $\boldsymbol{\theta}$ are known up to measurement errors $\delta\boldsymbol{p}$ and $\delta\boldsymbol{\theta}$, respectively. Moreover, the twist $\boldsymbol{t}$ is input into the control software of the robot with an unavoidable error $\delta\boldsymbol{t}$.

In solving (16.1) for $\dot{\boldsymbol{\theta}}$ with floating-point arithmetic, the computed value is contaminated with a roundoff error $\delta\dot{\boldsymbol{\theta}}$. The *relative error* in the computed $\dot{\boldsymbol{\theta}}$ is related to the relative errors in the architecture parameters and posture variables by the relation [16.30]

$$\frac{\|\delta\dot{\boldsymbol{\theta}}\|}{\|\dot{\boldsymbol{\theta}}\|} \le \kappa(\mathbf{J})\left(\frac{\|\delta\boldsymbol{p}\|}{\|\boldsymbol{p}\|} + \frac{\|\delta\boldsymbol{\theta}\|}{\|\boldsymbol{\theta}\|} + \frac{\|\delta\boldsymbol{t}\|}{\|\boldsymbol{t}\|}\right) \,, \tag{16.13}$$

where $\boldsymbol{p}$ and $\boldsymbol{\theta}$ represent the (unknown) actual values of these vectors and $\boldsymbol{t}$ the nominal value of the twist.

Nevertheless, the foregoing discussion applies to tasks involving either positioning or orientation, but not both. Most frequently, robotic tasks involve both positioning and orientation, which leads to Jacobian matrices with entries bearing disparate physical units, the consequence being that the Jacobian singular values have disparate units as well. Indeed, singular values associated with positioning bear units of length, whereas those associated with orientation are dimensionless. As a consequence, it is impossible to either order all singular values from smallest to largest or to add them all.

To cope with the issue of disparate units in the Jacobian entries, and to allow for the computation of the Jacobian condition number, the concept of *characteristic length* has been proposed [16.9]. The characteristic length L has been defined as the length by which the entries of the Jacobian with units of length are to be divided to render the Jacobian condition number a minimum at an optimum posture. This definition, while sound, lacks an immediate geometric interpretation, which has made its adoption in the robotics community difficult. In order to provide for a geometric interpretation, the concept of *homogeneous space* was recently introduced [16.33]. Using this concept, the robot architecture is designed in a *dimensionless space*, with points whose coordinates are dimensionless real numbers. In this way, the six Plücker coordinates [16.34] of lines are all dimensionless, and hence the columns

of the robot Jacobian matrix, comprising the Plücker coordinates of the revolute axes, are dimensionless as well. As a consequence, the Jacobian singular values are all dimensionless, and the Jacobian condition number can be defined. Once the robotic architecture has been designed for minimum condition number, under geometric constraints on link-length ratios and angles between neighboring revolute axes, for example, the maximum reach of the robot can be calculated. The maximum reach r will thus be a dimensionless quantity. When this quantity is compared with the prescribed maximum reach R, with units of length, the characteristic length is computed as the ratio $L = R/r$.

16.3.2 Dynamics-Based Local Performance Evaluation

Since motions are caused by forces and torques acting on rigid bodies, it seems reasonable to formulate performance indices that take into account the inertial properties of the mechanism. *Asada* [16.35] defines the *generalized-inertia ellipsoid* (GIE) as the ellipsoid described by the product $\mathbf{G} = \mathbf{J}^{-\mathrm{T}}\mathbf{M}\mathbf{J}^{-1}$, where $\mathbf{M}$ denotes the inertia matrix of the manipulator. This ellipsoid is that with semiaxes given by the singular values of the foregoing product. *Yoshikawa* [16.36], in turn, defines a corresponding dynamic manipulability measure as $\det[\mathbf{J}\mathbf{M}^{-1}(\mathbf{J}\mathbf{M}^{-1})^{\mathrm{T}}]$. Physically these concepts represent two distinct phenomena. Suppose the robot is viewed as an input–output device, which, given a joint torque, produces an acceleration at the EE. Yoshikawa's index measures the uniformity of this torque–acceleration gain, whereas Asada's GIE characterizes the inverse of this gain. If a human operator were holding the EE and attempting to move it about, the GIE would measure the resistance of the robot to this EE motion.

Other measures that attempt to capture robot performance as a function of the dynamics can be cited: *Voglewede* and *Ebert-Uphoff* [16.37] propose performance indices based on joint stiffness and link inertia, with the aim of establishing a distance to singularity for any robot posture, while *Bowling* and *Khatib* [16.38] propose a general framework for capturing the dynamic capability of a general robot manipulator that includes the velocity and acceleration characteristics of the EE, taking into account factors such as torque and the velocity limits of the actuators.

16.3.3 Global Dexterity Measures

The above measures are local in the sense that they characterize the dexterity of a robot at a given posture. Local measures are useful for applications ranging from redundancy resolution to workpiece positioning, but for design applications a global measure may be more desirable. One straightforward way of extending local measures to global ones is to integrate them over the allowable joint space. In [16.39] *Gosselin* and *Angeles* integrate the Jacobian condition number over the workspace to define a global measure, thereby producing a *global conditioning index*. For the simpler cases of planar positioning and spherical manipulators, the global conditioning index was found to coincide with its local counterpart.

16.3.4 Closed-Chain Dexterity Indices

The formulation of dexterity for closed chains presents a number of subtleties. The first obvious difference is that the joint configuration space of a closed chain will no longer be a flat space; in the general case it will be a curved multidimensional surface embedded in a higher-dimensional (typically flat) space. Also, dual to the open chain case, the forward kinematics for closed chains is generally more difficult to solve than the inverse kinematics, with the possibility of multiple solutions. Another important difference is that only a subset of the joints may be actuated, and that the number of actuated joints may even exceed the mechanism's kinematic degrees of freedom.

Several coordinate-based formulations for closed-chain dexterity have been proposed for specific mechanisms [16.40] and for cooperating robot systems whose joints are all assumed to be actuated [16.41, 42], at least some of which have led to apparently paradoxical results [16.43, 44]. Because of the nonlinear characteristics unique to closed-chain mechanisms discussed above, particular care must be exercised in formulating coordinate-based dexterity measures.

Another recent line of work has attempted a coordinate-invariant differential-geometric formulation of dexterity for closed chains. In this framework the joint and EE configuration spaces are regarded as Riemannian manifolds with an appropriate choice of Riemannian metric, with the choice of joint space metric reflecting the characteristics of the joint actuators. The previous notions of ellipsoid developed for serial chains can be extended to general closed chains, containing both passive and active joints, and possibly redundantly actuated [16.45, 46].

16.3.5 Alternative Dexterity-Like Measures

The various definitions of dexterity discussed above all refer to the same qualitative feature of the ability of a robot to move and apply forces in arbitrary directions. A different viewpoint is taken in the work of

Liégeois [16.47] and *Klein* and *Huang* [16.48], where dexterity is quantified in terms of joint-range availability. The driving motivation here lies in that most robots have joint limits; therefore, one should minimize the possibility of a joint reaching a stop.

Hollerbach [16.49] takes an alternative approach to designing a redundant seven revolute 7R manipulator, by considering: (a) elimination of internal singularities, (b) ability to avoid obstacles in the workspace, (c) the solvability of the kinematic equations, and (d) mechanical constructability. Based on these four criteria, he derived a particular 7R design with the morphology of the human arm, i. e., composed of two spherical joints defining the shoulder and the wrist plus an intermediate revolute playing the role of the elbow. Now, while a redundant architecture should remain fully capable of performing 6-DOF tasks upon locking one joint, the architecture reported in this reference may end up by losing this capability if the elbow joint is locked.

From a control perspective, *Spong* [16.50] shows that, if the inertia matrix of a manipulator has a vanishing Riemannian curvature, there exists a set of coordinates in which the equations of motion assume a particularly simple form. The curvature of the inertia matrix also reflects the sensitivity of the dynamics to certain robot parameters. Minimizing the curvature, therefore, is another possible criterion for robot design.

16.4 Other Performance Indices

16.4.1 Acceleration Radius

Another measure that attempts to capture the dynamic capabilities of a robotic manipulator is the *acceleration radius*. Initially proposed by *Graettinger* and *Krogh* in [16.51], the acceleration radius measures the minimum acceleration capability of the EE, in arbitrary directions, for the given torque bounds on the actuators. Specifically, given the dynamics equations for a serial chain in the form

$$\boldsymbol{\tau} = \mathbf{M}(\boldsymbol{\theta})\ddot{\boldsymbol{\theta}} + \mathbf{C}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}})\dot{\boldsymbol{\theta}} \,, \tag{16.14}$$

in which $\mathbf{M}$ is the robot *mass matrix* – also known as the *inertia matrix* – in *joint space*, and $\mathbf{C}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}})$ is the matrix mapping the joint-rate vector to the vector of Coriolis and centrifugal forces in the same space. Moreover, the actuators are assumed to have joint torque limits of the form

$$\boldsymbol{\tau}_{\min} \leq \boldsymbol{\tau} \leq \boldsymbol{\tau}_{\max} \,, \tag{16.15}$$

where the lower and upper limits $\boldsymbol{\tau}_{\min}, \boldsymbol{\tau}_{\max} \in \mathbb{R}^n$ are constant or functions of the manipulator posture $\boldsymbol{\theta}$. The EE *twist rate*, denoted by $\dot{\boldsymbol{t}}$, can be expressed as

$$\dot{\boldsymbol{t}} = \mathbf{J}(\boldsymbol{\theta})\ddot{\boldsymbol{\theta}} + \dot{\mathbf{J}}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}})\dot{\boldsymbol{\theta}} \,, \tag{16.16}$$

where $\dot{\mathbf{J}}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}})$ is the Jacobian time-derivative. The Jacobian $\mathbf{J}$ was introduced in (16.1).

Under the assumption that $\mathbf{J}(\boldsymbol{\theta})$ is nonsingular, one can write

$$\ddot{\boldsymbol{\theta}} = \mathbf{J}(\boldsymbol{\theta})^{-1}\dot{\boldsymbol{t}} - \boldsymbol{J}(\boldsymbol{\theta})^{-1}\dot{\mathbf{J}}(\boldsymbol{\theta}, \boldsymbol{t})\dot{\boldsymbol{\theta}} \,. \tag{16.17}$$

Substituting the above expression into the dynamic equations (16.14) leads to

$$\boldsymbol{\tau}(\boldsymbol{\theta}, \boldsymbol{t}, \dot{\boldsymbol{t}}) = \mathbf{M}'(\boldsymbol{\theta})\dot{\boldsymbol{t}} + \mathbf{C}'(\boldsymbol{\theta}, \boldsymbol{t})\boldsymbol{t} \,, \tag{16.18}$$

where

$$\mathbf{M}'(\boldsymbol{\theta}) = \mathbf{M}(\boldsymbol{\theta})\mathbf{J}(\boldsymbol{\theta})^{-1}$$
$$\mathbf{C}'(\boldsymbol{\theta}, \boldsymbol{t}) = [\mathbf{C}(\boldsymbol{\theta}, \boldsymbol{t}) - \mathbf{M}(\boldsymbol{\theta})\mathbf{J}(\boldsymbol{\theta})^{-1}\dot{\mathbf{J}}(\boldsymbol{\theta}, \boldsymbol{t})]\mathbf{J}^{-1}(\boldsymbol{\theta}) \,.$$

For a given state $(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}})$, the linear torque bounds (16.15) now define a polytope in the twist-rate space. *Graettinger* and *Krogh* [16.51] define the acceleration radius to be the largest sphere centered at the origin that is contained in this polytope; its radius reflects the minimal guaranteed EE acceleration in arbitrary directions. This concept is applied to measure the EE acceleration capability of a manipulator, as well as to determine the actuator sizes for achieving a desired acceleration radius. *Bowling* and *Khatib* [16.38] generalize this concept to capture both force and acceleration capabilities of the EE, with a view to quantifying the worst-case dynamic performance capability of a manipulator.

16.4.2 Elastostatic Performance

Elastostatic performance refers to the robotic system's response to the applied load – force and moment, i. e., wrench – under static equilibrium. This response may be measured in terms of the *stiffness* of the manipulator, which determines the translation and angular deflections when the EE is subjected to an applied wrench.

Robot deflections have two sources: link and joint deflection. In the presence of long links, as in the space

robot Canadarm2, link compliance is a major source of deflection. However, in the majority of today's serial robots, the main source of deflection occurs at the joints.

In this chapter, we assume that the manipulator links are rigid, and model the joints as linearly elastic torsional springs. The more complex problem of link flexibility is studied in detail in Chap. 11. That is, for the elastostatic model we base our analysis on the assumption that, under a positioning task, the joints are locked at a certain posture $\boldsymbol{\theta}_0$, while the EE is subject to a perturbation wrench $\Delta\boldsymbol{w}$ that is balanced by an elastic joint torque $\Delta\boldsymbol{\tau}$. Under these conditions, $\Delta\boldsymbol{\theta}$ and $\Delta\boldsymbol{\tau}$ obey the well-known linear relation

$$\mathbf{K}\Delta\boldsymbol{\theta} = \Delta\boldsymbol{\tau} \ , \tag{16.19}$$

in which $\mathbf{K}$ is the *stiffness matrix* in joint space at the given posture. Moreover, the matrix $\mathbf{K}$ is diagonal, with its entries equal to the torsional stiffnesses of the corresponding joints, $\mathbf{K}$ is therefore *posture independent*, i. e., constant throughout the whole robot workspace. Moreover, since all joints exhibit a finite, nonzero stiffness, $\mathbf{K}$ is invertible, its inverse $\mathbf{C}$ being termed the *compliance matrix*. We can thus express the inverse relation of (16.19) as

$$\Delta\boldsymbol{\theta} = \mathbf{C}\Delta\boldsymbol{\tau} \ . \tag{16.20}$$

It should be apparent that $\Delta\boldsymbol{\theta}$ and $\Delta\boldsymbol{\tau}$ have an *incremental* nature, for both are measured from the equilibrium posture, at which $\Delta\boldsymbol{\tau}$ and $\Delta\boldsymbol{\theta}$ vanish.

On the issue of stiffness matrix, *Griffis* and *Duffy* [16.52] proposed a mapping from an incremental rigid-body displacement $\Delta\boldsymbol{x}$ into an incremental wrench $\Delta\boldsymbol{w}$ that turned out to be nonsymmetric. The concept behind this mapping was formalized by *Howard*, *Zefran* and *Kumar* [16.53] by means of Lie algebra. However, in the foregoing papers, $\Delta\boldsymbol{x}$ and $\Delta\boldsymbol{w}$ turn out to be incompatible in the sense that their reciprocal product does not yield incremental work – the point at which $\Delta\boldsymbol{x}$ is defined is distinct from that at which $\Delta\boldsymbol{w}$ is applied. For this reason, the array representation of the same mapping cannot be, properly speaking, termed a stiffness matrix.

For a constant magnitude of $\Delta\boldsymbol{\tau}$, the deflection attains its maximum value in the direction of the eigenvector associated with the maximum eigenvalue of $\mathbf{C}$ or, equivalently, with the minimum eigenvalue of $\mathbf{K}$, denoted by $\kappa_{\min}$. In terms of elastostatic performance, we aim to (a) make the maximum deflection a minimum, i. e., we want to maximize $\kappa_{\min}$, and (b) make the magnitude of the deflection $||\Delta\boldsymbol{\theta}||$ as insensitive as possible to changes in the direction of the applied load $\Delta\boldsymbol{\tau}$. This can be done by rendering $\kappa_{\min}$ as close as possible to $\kappa_{\max}$. The first aim is associated with the stiffness constants, i. e., the higher the constants the lower the deflections. The later, however, is associated with the concept of isotropy, the ideal case being when all the eigenvalues of $\mathbf{K}$ are identical, which means that $\mathbf{K}$ is isotropic. Due to the *pyramidal effect* of serial robots, in which downstream motors carry their upstream counterparts, the joint stiffness is bound to be largest for the proximal joints. Hence an isotropic stiffness matrix is impossible for serial robots.

Notice that (16.19) and (16.20) may also be formulated in the task space, i. e.,

$$\mathbf{K}_{\mathrm{C}}\Delta\boldsymbol{x} = \Delta\boldsymbol{w} \ , \tag{16.21}$$

where $\Delta\boldsymbol{x} \equiv \boldsymbol{t}\Delta t$, with Δt denoting a *small* time interval producing a correspondingly *small* change $\Delta\boldsymbol{x}$ in the pose of the EE. That is,

$$\Delta\boldsymbol{x} = \mathbf{J}\dot{\boldsymbol{\theta}}\,\Delta t = \mathbf{J}\Delta\boldsymbol{\theta} \ , \tag{16.22}$$

which is a linear transformation of the joint-vector increment into the pose-increment vector. We show next that *the stiffness matrix is not frame invariant*. This means that, under the linear transformation from joint to Cartesian coordinates, *the stiffness matrix does not obey a similarity transformation*. For quick reference, we recall below the definition of similarity transformation: if $\boldsymbol{y} = \mathbf{L}\boldsymbol{x}$ is a linear transformation of $\mathbb{R}^n$ into itself, and we introduce a change of vector basis, $\boldsymbol{x}' = \mathbf{A}\boldsymbol{x}$, $\boldsymbol{y}' = \mathbf{A}\boldsymbol{y}$, then $\mathbf{L}$ changes to $\mathbf{L}'$, which is given by

$$\mathbf{L}' = \mathbf{A}\mathbf{L}\mathbf{A}^{-1} \ . \tag{16.23}$$

The above transformation of any vector of $\mathbb{R}^n$ into another one of the same space, and of a matrix $\mathbf{L}$ into $\mathbf{L}'$, as given by (16.23), is termed a *similarity transformation*. Notice that $\mathbf{A}$ is guaranteed to be invertible, as it represents a change of coordinate frame.

Now, under the change of coordinates given by (16.22), (16.21) leads to

$$\mathbf{K}_{\mathrm{C}}\mathbf{J}\Delta\boldsymbol{\theta} = \mathbf{J}^{-\mathrm{T}}\Delta\boldsymbol{\tau} \ , \tag{16.24}$$

where we have used the kinetostatic relation [16.9]

$$\mathbf{J}^{\mathrm{T}}\Delta\boldsymbol{w} = \Delta\boldsymbol{\tau} \ .$$

The exponent $-\mathrm{T}$ denotes the transpose of the inverse or, equivalently, the inverse of the transpose. Upon multiplication of both sides of (16.24) by $\mathbf{J}^{\mathrm{T}}$ from the left, we end up with

$$\mathbf{J}^{\mathrm{T}}\mathbf{K}_{\mathrm{C}}\mathbf{J}\Delta\boldsymbol{\theta} = \Delta\boldsymbol{\tau} \ . \tag{16.25}$$

If we now compare (16.19) with (16.25), we can readily derive the relations between the stiffness matrix $\mathbf{K}$ in joint space and the stiffness matrix $\mathbf{K}_C$ in Cartesian space

$$\mathbf{K} = \mathbf{J}^T\mathbf{K}_C\mathbf{J} \quad \text{and} \quad \mathbf{K}_C = \mathbf{J}^{-T}\mathbf{K}\mathbf{J}^{-1} , \tag{16.26}$$

which apparently is not a similarity transformation between $\mathbf{K}$ and $\mathbf{K}_C$. What this means is that the two matrices do not share the same set of eigenvalues and their eigenvectors are not related by the linear relation (16.22). As a matter of fact, if the robot is revolute-coupled, the entries of its stiffness matrix $\mathbf{K}$ all have units of Nm, i. e., of *torsional stiffness*, while the entries of $\mathbf{K}_C$ have disparate units. To show this, the Jacobian matrix, the inverse Jacobian and the two stiffness matrices are partitioned correspondingly into four 3×3 blocks, i. e., as

$$\mathbf{J} = \begin{pmatrix} \mathbf{J}_{11} & \mathbf{J}_{12} \\ \mathbf{J}_{21} & \mathbf{J}_{22} \end{pmatrix} , \quad \mathbf{J}^{-1} = \begin{pmatrix} \mathbf{J}'_{11} & \mathbf{J}'_{12} \\ \mathbf{J}'_{21} & \mathbf{J}'_{22} \end{pmatrix} ,$$
$$\mathbf{K} = \begin{pmatrix} \mathbf{K}_{11} & \mathbf{K}_{12} \\ \mathbf{K}_{12}^T & \mathbf{K}_{22} \end{pmatrix} , \quad \mathbf{K}_C = \begin{pmatrix} \mathbf{K}_{C11} & \mathbf{K}_{C12} \\ \mathbf{K}_{C12}^T & \mathbf{K}_{C22} \end{pmatrix} .$$

Furthermore, in light of the definition of the twist, (16.2), the two upper blocks of $\mathbf{J}$ are dimensionless, while its two lower blocks have units of length [16.9]. As a consequence, the two left blocks of $\mathbf{J}^{-1}$ are dimensionless, while its two right blocks have units of inverse length. Now, the blocks of $\mathbf{K}_C$ are computed from the corresponding relation in (16.26), thereby obtaining

$$\begin{aligned}
\mathbf{K}_{C11} &= \mathbf{J}_{11}'^T(\mathbf{K}_{11}\mathbf{J}'_{11} + \mathbf{K}_{12}\mathbf{J}'_{21}) \\
&\quad + \mathbf{J}_{21}'^T(\mathbf{K}_{12}^T\mathbf{J}'_{11} + \mathbf{K}_{22}\mathbf{J}'_{21}) \\
\mathbf{K}_{C12} &= \mathbf{J}_{11}'^T(\mathbf{K}_{11}\mathbf{J}'_{12} + \mathbf{K}_{12}\mathbf{J}'_{22}) \\
&\quad + \mathbf{J}_{21}'^T(\mathbf{K}_{12}^T\mathbf{J}'_{12} + \mathbf{K}_{22}\mathbf{J}'_{22}) \\
\mathbf{K}_{C21} &= \mathbf{K}_{C12}^T \\
\mathbf{K}_{C22} &= \mathbf{J}_{12}'^T(\mathbf{K}_{11}\mathbf{J}'_{12} + \mathbf{K}_{12}\mathbf{J}'_{22}) \\
&\quad + \mathbf{J}_{22}'^T(\mathbf{K}_{12}^T\mathbf{J}'_{12} + \mathbf{K}_{22}\mathbf{J}'_{22}) .
\end{aligned}$$

It is now apparent that: $\mathbf{K}_{C11}$ has entries with units of Nm, i. e., of torsional stiffness; $\mathbf{K}_{C12}$ and $\mathbf{K}_{C21}$ have entries with units of N; and $\mathbf{K}_{C22}$ has entries with units of N/m, i. e., of translational stiffness.

The outcome of the foregoing discussion is that a norm for $\mathbf{K}$ is possible, but not one for $\mathbf{K}_C$, unless of course a characteristic length is introduced to render all the entries of $\mathbf{K}_C$ dimensionally homogeneous. A norm of a matrix is useful as it indicates *how large* the entries of the matrix are. We would like to characterize how stiff a robot is in both joint and Cartesian spaces. In joint space we could adopt any norm, but notice that the 2-norm, introduced in (16.8), would be misleading, as this norm would assign the value of the strongest joint to the overall robot stiffness. A more suitable norm would be the weighted Frobenius norm, introduced in (16.11), which would assign the rms value of the joint stiffnesses to the overall robot stiffness.

To design a robot optimally, we would therefore aim to maximize the Frobenius norm of its stiffness matrix in joint space, while observing constraints on the robot weight, as stiffer joints lead to heavier joints if the same material is used for all joints.

16.4.3 Elastodynamic Performance

For a general design problem, not only the kinetostatic and elastostatic performances, but also the elastodynamic performance have to be considered. In this regard, we introduce the assumptions of Sect. 16.4.2, with the added condition that inertia forces due to the link masses and moments of inertia are now taken into consideration.

The linearized model of a serial robot at the posture given by $\boldsymbol{\theta}_0$, if we neglect damping, is

$$\mathbf{M}\Delta\ddot{\boldsymbol{\theta}} + \mathbf{K}\Delta\boldsymbol{\theta} = \Delta\boldsymbol{\tau} , \tag{16.27}$$

in which $\mathbf{M}$ is the $n \times n$ positive-definite mass matrix introduced in Sect. 16.4.1, while $\mathbf{K}$ was defined in Sect. 16.4.2 as the $n \times n$ positive-definite stiffness matrix in joint space. Both $\mathbf{K}$ and $\mathbf{M}$ were defined in joint-space coordinates, $\Delta\boldsymbol{\theta}$, representing the vector of joint-variable elastic displacements, as in Sect. 16.4.2. These displacements are produced when, as the joints are all locked at a value $\boldsymbol{\theta}_0$ and thereby become ideal linear springs, the robot is subject to a perturbation $\Delta\boldsymbol{\tau}$, to nonzero initial conditions, or to a combination of both.

Under *free vibration*, i. e., under a motion of the system (16.27) caused by nonzero initial conditions and a zero excitation $\Delta\boldsymbol{\tau}$, the foregoing equation can be solved for $\Delta\ddot{\boldsymbol{\theta}}$

$$\Delta\ddot{\boldsymbol{\theta}} = -\mathbf{D}\Delta\boldsymbol{\theta} , \quad \mathbf{D} \equiv \mathbf{M}^{-1}\mathbf{K} , \tag{16.28}$$

in which the matrix $\mathbf{D}$ is known as the *dynamic matrix*. This matrix determines the behavior of the system under consideration, as its eigenvalues are the *natural frequencies* of the system and its eigenvectors the *modal vectors*. Let $\{\omega_i\}_1^n$ and $\{\boldsymbol{f}_i\}_1^n$ denote the sets of eigenvalues and the corresponding eigenvectors of $\mathbf{D}$, respectively. Under the initial conditions $[\Delta\boldsymbol{\theta}(0), \Delta\dot{\boldsymbol{\theta}}(0)]^T$, in which $\Delta\boldsymbol{\theta}(0)$ is proportional to the i-th

eigenvector of $\mathbf{D}$ and $\Delta\dot{\boldsymbol{\theta}}(0) = \mathbf{0}$, the ensuing motion is of the form $\Delta\boldsymbol{\theta}(t) = \Delta\boldsymbol{\theta}(0)\cos\omega_i t$ [16.54].

Furthermore, under the change of variables given by (16.22), the model (16.27) changes to

$$\mathbf{M}\mathbf{J}^{-1}\Delta\ddot{\boldsymbol{x}} + \mathbf{K}\mathbf{J}^{-1}\Delta\boldsymbol{x} = \mathbf{J}^{\mathrm{T}}\Delta\boldsymbol{w}\,.$$

If we now multiply both sides of the above equation by $\boldsymbol{J}^{-\mathrm{T}}$, we obtain the elastodynamic model (16.27) in Cartesian coordinates, namely

$$\mathbf{J}^{-\mathrm{T}}\mathbf{M}\mathbf{J}^{-1}\Delta\ddot{\boldsymbol{x}} + \mathbf{J}^{-\mathrm{T}}\mathbf{K}\mathbf{J}^{-1}\Delta\boldsymbol{x} = \Delta\boldsymbol{w}\,,$$

in which the first matrix coefficient is the mass matrix $\mathbf{M}_{\mathrm{C}}$ in Cartesian coordinates, and the second is identified as $\mathbf{K}_{\mathrm{C}}$, which was introduced in (16.26), i. e.,

$$\mathbf{M}_{\mathrm{C}} \equiv \mathbf{J}^{-\mathrm{T}}\mathbf{M}\mathbf{J}^{-1}\,. \tag{16.29}$$

The elastodynamic model in Cartesian coordinates thus takes the form

$$\mathbf{M}_{\mathrm{C}}\Delta\ddot{\boldsymbol{x}} + \mathbf{K}_{\mathrm{C}}\Delta\boldsymbol{x} = \Delta\boldsymbol{w}\,. \tag{16.30}$$

Again, by virtue of the transformation (16.29), it is apparent that the mass matrix, like its stiffness counterpart, is not invariant under a change of coordinates. Moreover, in a revolute-coupled robot, all the entries of $\mathbf{M}$ have units of kg m^2; however, the entries of $\mathbf{M}_{\mathrm{C}}$ have disparate units. An analysis similar to that conducted in Sect. 16.4.2 for the stiffness matrix in Cartesian space reveals that, if $\mathbf{M}_{\mathrm{C}}$ is partitioned into four 3×3 blocks, then its upper-left block has units of moment of inertia, its lower-right block has units of mass, while its off-diagonal blocks have units of kg m.

Correspondingly, the dynamic matrix in Cartesian coordinates becomes

$$\mathbf{D}_{\mathrm{C}} = \mathbf{M}_{\mathrm{C}}^{-1}\mathbf{K}_{\mathrm{C}}\,. \tag{16.31}$$

It is now a simple matter to prove that *the dynamic matrix is invariant under a change of coordinates*. Indeed, if we substitute transformations (16.26) and (16.29) into (16.31), we obtain

$$\mathbf{D}_{\mathrm{C}} = \mathbf{J}\mathbf{M}^{-1}\mathbf{J}^{\mathrm{T}}\mathbf{J}^{-\mathrm{T}}\mathbf{K}\mathbf{J}^{-1} = \mathbf{J}\mathbf{M}^{-1}\mathbf{K}\mathbf{J}^{-1}\,,$$

in which the dynamic matrix $\mathbf{D}$ in joint coordinates can be readily identified in the final expression, and hence,

$$\mathbf{D}_{\mathrm{C}} = \mathbf{J}\mathbf{D}\mathbf{J}^{-1}\,, \tag{16.32}$$

which shows that $\mathbf{D}_{\mathrm{C}}$ is indeed a similarity transformation of $\mathbf{D}$. As a consequence, the dynamic matrix is indeed invariant under a change of coordinates, the two matrices thus sharing the same sets of eigenvalues, their eigenvectors being related by the same similarity transformation. That is, if the modal vectors in joint space – the eigenvectors of $\mathbf{D}$ – are denoted by $\{\boldsymbol{f}_i\}_1^n$ and the modal vectors in Cartesian space are denoted by $\{\boldsymbol{g}_i\}_1^n$, then these two sets are related by

$$\boldsymbol{g}_i = \mathbf{J}\boldsymbol{f}_i\,, \quad i = 1,\ldots,n\,. \tag{16.33}$$

Therefore, the natural frequencies of the elastodynamic model are the same, regardless of the space in which they are calculated, while their natural modes of vibration change under a similarity transformation.

Under an excitation of the form $\Delta\boldsymbol{\tau} = \boldsymbol{\theta}_0\cos\omega t$ and zero initial conditions, the system is known to respond with a harmonic motion of frequency ω and magnitude that depends on both ω and the system *frequency spectrum* $\{\omega_i\}_1^n$ [16.54]. When ω equals one of the natural frequencies of the system, the response magnitude grows unbounded, a phenomenon known as *resonance*. For this reason, when designing a robot, it is imperative that its frequency spectrum does not involve any of the expected operation frequencies, which can be achieved by designing the robot with stiffness and mass matrices that yield a frequency spectrum outside of the frequency range of the operation conditions.

This design task is not straightforward. Indeed, while the stiffness matrix in joint space is constant, the mass matrix is posture dependent, i. e., $\mathbf{M} = \mathbf{M}(\boldsymbol{\theta})$. Because of this feature, the elastodynamic design of a robot is bound to be iterative: the design can be conducted for a *straw-man task*, i. e., a typical task in the target applications, thus defining a set of postures that lead in turn to a set of mass-matrix values. Then, the frequency spectrum for all these postures can be designed to lie above the frequency range of the straw-man task. Since the robot will eventually be commanded to execute other tasks than the straw-man task, simulation of alternative tasks is required to ensure that the design is safe from a resonance viewpoint.

16.5 Other Robot Types

Serial robots were fully developed, to a great extent, in the nineties, with SCARA systems capable of executing the standard manipulation task defined in Sect. 16.2.1 at the rate of two cycles per second. Other, more sophisticated robots started appearing in the 1980s. Parallel robots were a natural extension of serial robots. They

presented new challenges to researchers, as their forward kinematics turned out to be quite more complex than the inverse kinematics of serial robots. Nowadays a few parallel and hybrid, i. e., serial-parallel robots, are found in applications in industry and other environments. A remarkable type of robot that emerged in the 1990s is *humanoids*, with their own design and control challenges. Flying and swimming robots are also developing fast. An outline of the design challenges brought about by these new types of robots is included below.

16.5.1 Robots Mounted on a Fixed Base: Parallel-Kinematics Machines

Under the umbrella *parallel-kinematics machines* (PKMs) are included robots comprising one base plate (BP), one moving plate (MP) and a set of limbs that couple the two. The earliest robot of this kind that attracted the attention of researchers is what is known as the *Gough–Stewart platform* (GSP), consisting of six limbs, each of the SPU type, where S, P and U stand for spherical, prismatic and universal joint, respectively. In fact, the first two belong to what is known as the six lower kinematic pairs [16.3] – the other four are R, H, C, and F, which are termed revolute, helical, cylindrical and planar pairs. The U joint is a concatenation of two R pairs with axes intersecting at right angles. Three of these six pairs, R, P and H, allow for 1-DOF motion between the two links that they couple; C allows for 2-DOF, while S and F allow for three. Each limb of the GSP thus allows for 6-DOF of the MP with respect to the BP.

The first and foremost task when designing a PKM is the choice of both the type and the number of limbs. As to the latter, it varies from two to six. Apparently, the DOF of a limb should be at least that of the PKM under design. The early PKMs studied, mainly in the 1980s, were invariably 6-DOF systems. *Clavel* [16.6] patented in 1990 what may be the first PKM with *reduced mobility*, i. e., with a DOF = 3, namely, a robot that allows pure translations of its MP. Spherical robots, like *Gosselin*'s *Agile Eye* [16.55], are also of reduced mobility. Different from Clavel's Delta, all links of spherical linkages undergo spherical motion, rotating about the same point – the center of the sphere. Delta is supplied with links that undergo other motions than those of their MP, purely translational.

Other PKMs with reduced mobility appeared later, the best known being the H4, a 4-DOF system, patented in 2001 [16.56], and designed to implement Schönflies displacements (Sect. 16.2.1) of its MP. The foregoing patent came as a response to the need of ever faster pick-and-place robots for the packaging and the electronics industry sectors, when it was apparent that the SCARA systems of the mid eighties had reached their limits because of their serial architecture. The H4 is manufactured by Adept Technology, Inc. under the name *Adept Quattro s650HS parallel robot*.

Motivated by the growing interest in fast robots with reduced mobility, two approaches to the design of PKMs with reduced mobility have been put forth: *screw theory* [16.57] and *group theory* [16.58]. By far, the most favored approach is the former, as screw theory is well established within the community of robot mechanicists. However, group theory offers features not found in screw theory, which is *local* in that its scope is limited to the *velocity level*. Group theory is applicable at the *displacement level*. What is at stake in this context is the *topology* of the robot, namely, the number of joints, links, limbs and the type of joints, what *Hartenberg* and *Denavit* [16.3] term the *number synthesis* and the *type synthesis*. These two activities correspond to the *qualitative synthesis* underlying the design process, which appears at the *conceptual* stage, as opposed to its *quantitative* counterpart; the latter appears at the *embodiment* and the *detail* stages of the design process [16.4]. Recently, a design paradigm based on the concept of complexity has been proposed [16.59], aimed at assessing the complexity of a linkage design variant based on the types of its joints.

As an extension of the Delta robot, ABB Robotics developed the *Flexpicker IRB360*, a hybrid system that comprises a Delta robot in series with a revolute joint at its MP that adds one fourth DOF to the purely translational motion of the Delta. The result is, thus, a hybrid Schönflies motion generator. ABB's robot is used in the food-processing industry.

A major bottleneck in the development of fast robots with reduced mobility is their structural stiffness. With the packaging and electronics industry requiring ever faster pick-and-place robots, capable of speeds greater than 3 cycles/s, high excitation frequencies are generated, which calls for structures much stiffer than those found in other applications areas. The ideal fast robot should have a parallel structure whose mass is negligible when compared to that of its payload and, at the same time, highly stiff. As the robots of interest are intended for the production of Schönflies displacements, any motion of their MP outside of this subgroup, namely, any tilt of the MP, is considered parasitical and should be of as small an amplitude as possible. This calls for a structure that exhibits a high rotational stiffness about any horizontal axis – under the assumption that the only rotational motion of the MP is about a vertical axis, which is usually the case. The challenge here is how to characterize this stiffness. A simple elastodynamic model of the structure is derived by assuming

that the limbs provide a linearly elastic suspension for the MP, which is assumed to be rigid. The structure can be regarded as a *generalized spring* [16.60], namely, a linearly elastic suspension that is fixed to one stationary base at one end and carries a moving rigid body at another end. Moreover, the suspension allows full 6-DOF *small-amplitude* displacements of the body. What is meant by the latter is rotations that entail a *small* angle – as small as to allow for an approximation of the type $\sin\theta \approx \theta$, with θ denoting the angle – and translations small with respect to the dimensions of the body. The motion of the MP can then be represented using a unit screw [16.57] multiplied by a *small* angle Θ. Under these conditions, the Cartesian stiffness matrix is of 6×6, given that the screw is of dimension 6.

In characterizing the rotational stiffness of PKMs, one may think of resorting to the eigenvalues and eigenvectors of their stiffness matrix. There are some issues with this approach, however, for, in light of the different units of the four blocks of this matrix (Sect. 16.4.2) the eigenvalue problem associated with the matrix is meaningful only in a *generalized sense* [16.61], as explained presently. Let us recall the 6×6 Cartesian stiffness matrix $\mathbf{K}_C$, introduced in Sect. 16.4.2, and let us denote with $\boldsymbol{k}_i$ its i-th eigenvector, a unit screw, and with κ_i its corresponding eigenvalue. The generalized eigenvalue problem associated with $\mathbf{K}_C$ now can be stated as

$$\mathbf{K}_C\boldsymbol{k}_i = \kappa_i\boldsymbol{\Gamma}\boldsymbol{k}_i,\ \boldsymbol{k}_i \equiv \begin{pmatrix}\boldsymbol{e}_i\\ \boldsymbol{\mu}_i\end{pmatrix},\ i=1,\ \ldots\ 6\,, \tag{16.34}$$

in which $\boldsymbol{e}_i$ is the unit vector denoting the direction of the axis of the i-th eigenscrew, $\boldsymbol{\mu}_i$ is the *moment* of the axis with respect to the origin and $\boldsymbol{\Gamma}$ is a permutation matrix that exchanges the top and the bottom blocks of a screw array, these two items being defined as

$$\boldsymbol{\Gamma} = \begin{pmatrix}\mathbf{O} & \mathbf{1}\\ \mathbf{1} & \mathbf{O}\end{pmatrix},\ \boldsymbol{\mu}_i = \boldsymbol{e}_i\times\boldsymbol{p}_i + p_i\boldsymbol{e}_i\,, \tag{16.35}$$

for $i=1,\ \ldots\ 6$, $\mathbf{O}$ and $\mathbf{1}$ denoting the 3×3 zero and identity matrices, respectively, while $\boldsymbol{p}_i$ is the position vector of a point on the axis of the eigenscrew $\boldsymbol{k}_i$, while p_i is the *pitch* of $\boldsymbol{k}_i$. The moment of a line $\mathcal{L}$ can be interpreted as that of a unit force whose line of action coincides with $\mathcal{L}$. The generalized eigenvalue problem of (16.34) now leads to

$$\mathbf{A}_i\boldsymbol{k}_i = \mathbf{0},\ \mathbf{A}_i = \begin{pmatrix}\mathbf{K}_{rr} & \mathbf{K}_{rt}-\kappa_i\mathbf{1}\\ \mathbf{K}_{rt}^{\mathrm{T}}-\kappa_i\mathbf{1} & \mathbf{K}_{tt}\end{pmatrix}, \tag{16.36}$$

for $i=1,\ \ldots\ 6$. In light of its definition, (16.34), $\boldsymbol{k}_i \neq \mathbf{0}$, and hence, (16.36) must admit a non-trivial solution, which means that $\mathbf{A}_i$ must be singular, thereby leading to the characteristic equation of $\mathbf{K}_C$, namely,

$$\det\begin{pmatrix}\mathbf{K}_{rr} & \mathbf{K}_{rt}-\kappa\mathbf{1}\\ \mathbf{K}_{rt}^{\mathrm{T}}-\kappa\mathbf{1} & \mathbf{K}_{tt}\end{pmatrix} = 0\,, \tag{16.37}$$

where subscript i has been deleted from κ, for it is no longer needed. In light of the linear homogeneous form in κ of the foregoing 6×6 matrix, its determinant is a sextic polynomial in κ, the eigenvalue sought.

It is worth mentioning that the set of screws does not form, properly speaking, a vector space, for screws do not have a zero – they include a three-dimensional unit vector, which hence cannot vanish. By the same token, the set of screws does not admit a scalar product. Indeed, if two screw arrays are dot-multiplied, their product is meaningless, because it involves additions of quantities with disparate units. The pertinent concept is *reciprocity*: the *reciprocal product* of two unit screws, $\boldsymbol{s}_1$ and $\boldsymbol{s}_2$ is defined via matrix $\boldsymbol{\Gamma}$, introduced in (16.35), namely, $\boldsymbol{s}_1^{\mathrm{T}}\boldsymbol{\Gamma}\boldsymbol{s}_2$, which rightfully has units of length. When the reciprocal product of two unit screws vanishes, the screws are said to be *reciprocal* to each other. The physical significance of the concept lies in that a unit screw multiplied by an amplitude F with units of force produces a *wrench*; one multiplied by an amplitude Ω with units of frequency yields a twist. If the wrench happens to be a pure force with line of action intersecting the axis of the twist – the case of a force applied at a door whose line of action intersects the axis of the hinges – then wrench and twist are reciprocal to each other in that the power developed by the wrench on the door undergoing a *feasible* twist is zero.

Now we have a result that parallels one applicable to symmetric $n\times n$ matrices:

Theorem 16.1

The eigenvalues of $\mathbf{K}$ are real and the product $\kappa_i p_i$ is non-negative, while the eigenvectors $\{\boldsymbol{k}_i\}_1^6$ are mutually reciprocal.

its proof being found in [16.62] and [16.63].

Further, *decoupling* of the stiffness matrix means finding a new coordinate frame in which its two off-diagonal blocks vanish. Such a decoupling is not always possible, as shown below (See the *corrigenda* in [16.63]). It is known that the entries of the Cartesian stiffness matrix change according with a similarity transformation applicable to screws, which involves both a shift of origin and a change of the orientation of the original frame [16.64]. If the latter is labeled $\mathcal{A}$ and the new frame $\mathcal{B}$, then $\mathcal{A}$ is carried into $\mathcal{B}$ by means of a shift of the origin given by vector $\boldsymbol{d}$ and a rotation matrix $\mathbf{Q}$. In the sequel, the cross-product matrix

of a vector $\boldsymbol{v} \in \mathbb{R}^3$ will be needed. This is known to be a skew symmetric matrix $\mathbf{V}$ with the property

$$\mathbf{V}\boldsymbol{x} \equiv \boldsymbol{v} \times \boldsymbol{x}, \quad \forall\, \boldsymbol{x} \in \mathbb{R}^3, \quad \mathbf{V} \equiv \mathrm{CPM}(\boldsymbol{v}) \,. \tag{16.38}$$

Now let $\mathbf{D} \equiv \mathrm{CPM}(\boldsymbol{d})$, with $\boldsymbol{d}$ defined above as the displacement that takes the origin of $\mathcal{A}$ into that of $\mathcal{B}$. The components of a unit screw $\hat{\mathbf{s}}$ in $\mathcal{B}$ change to those in $\mathcal{A}$ according to the similarity transformation

$$[\hat{\mathbf{s}}]_{\mathcal{A}} = \mathbf{S}[\hat{\mathbf{s}}]_{\mathcal{B}}, \quad \mathbf{S} = \begin{pmatrix} \mathbf{Q} & \mathbf{O} \\ \mathbf{DQ} & \mathbf{Q} \end{pmatrix} , \tag{16.39}$$

the inverse of the 6×6 matrix $\mathbf{S}$ being

$$\mathbf{S}^{-1} = \begin{pmatrix} \mathbf{Q}^{\mathrm{T}} & \mathbf{O} \\ -\mathbf{Q}^{\mathrm{T}}\mathbf{D} & \mathbf{Q}^{\mathrm{T}} \end{pmatrix} . \tag{16.40}$$

Accordingly, the components of the stiffness matrix change, under the same similarity transformation, as [16.63]

$$[\mathbf{K}]_{\mathcal{A}} = \mathbf{\Gamma}\mathbf{S}\mathbf{\Gamma}[\mathbf{K}]_{\mathcal{B}}\mathbf{S}^{-1} \,, \tag{16.41}$$

which, were it not for matrix $\mathbf{\Gamma}$, looks like the similarity transformation of a $n \times n$ matrix. The presence of $\mathbf{\Gamma}$ is essential by virtue of the definition of the reciprocal product. Let, now,

$$[\mathbf{K}]_{\mathcal{A}} = \begin{pmatrix} \mathbf{K}'_{rr} & \mathbf{K}'_{rt} \\ (\mathbf{K}'_{rt})^{\mathrm{T}} & \mathbf{K}'_{tt} \end{pmatrix} , \; [\mathbf{K}]_{\mathcal{B}} = \begin{pmatrix} \mathbf{K}_{rr} & \mathbf{K}_{rt} \\ \mathbf{K}_{rt}^{\mathrm{T}} & \mathbf{K}_{tt} \end{pmatrix} . \tag{16.42}$$

The distinct blocks of the two matrices are related by

$$\begin{aligned} \mathbf{K}'_{rr} &= \mathbf{Q}(\mathbf{K}_{rr} - \mathbf{K}_{rt}\mathbf{D})\mathbf{Q}^{\mathrm{T}} + \mathbf{DQ}(\mathbf{K}_{rt}^{\mathrm{T}} - \mathbf{K}_{tt}\mathbf{D})\mathbf{Q}^{\mathrm{T}} \\ \mathbf{K}'_{rt} &= (\mathbf{QK}_{rt} + \mathbf{DQK}_{tt})\mathbf{Q}^{\mathrm{T}} \\ \mathbf{K}'_{tt} &= \mathbf{QK}_{tt}\mathbf{Q}^{\mathrm{T}} \end{aligned} \,,$$

the decoupling condition following upon setting $\mathbf{K}'_{rt} = \mathbf{O}$, which leads to

$$\mathbf{Q}^{\mathrm{T}}\mathbf{DQK}_{tt} = -\mathbf{K}_{rt} \,, \tag{16.43}$$

Now, since $\mathbf{D}$ is a cross-product matrix, it is singular – its null space is spanned by vector $\boldsymbol{d}$. Based on Sylvester's Theorem [16.31], which states that the rank of the product of two matrices is at most the smaller of the matrix ranks, the left-hand side of (16.43) is singular, and hence, the right-hand side must also be singular, which is the decoupling condition sought. If block $\mathbf{K}_{rt}$ is singular, then decoupling can be achieved by means of a shift of the origin, without changing the orientation of the frame, i. e., upon choosing $\mathbf{Q} = \mathbf{1}$ in (16.43). This result is summarized below:

Theorem 16.2

The stiffness matrix can be decoupled if and only if its off-diagonal block is singular. When this is the case, decoupling is achieved by a *similarity transformation* involving only a shift of origin.

The design criterion to apply here is, thus, to aim at a singular coupling block of the Cartesian stiffness matrix. This can be achieved by means of finite elements, but these work for one single robot posture. However, with the help of the concept of generalized spring, *Taghvaiepour* et al. [16.65] showed that it is possible to obtain the stiffness matrix of deformable links using that obtained with finite element analysis (FEA) at one particular posture, then transforming this matrix to obtain its counterpart at an alternate posture. The details on how to compute the displacement vector $\boldsymbol{d}$ that helps decouple the stiffness matrix are given in [16.63].

Once the stiffness matrix is decoupled, the eigenvalues of the rotational and translational blocks can be readily found, and assess the performance of the robot from the stiffness viewpoint. As a matter of fact, in the presence of a decoupled matrix, the characteristic equation (16.37) reduces to

$$\det \begin{pmatrix} \mathbf{K}_{rr} & -\kappa\mathbf{1} \\ -\kappa\mathbf{1} & \mathbf{K}_{tt} \end{pmatrix} = 0 \,, \tag{16.44}$$

Upon expansion, using the expression for the determinant of a block-matrix [16.66], the foregoing equation leads to

$$\det(\mathbf{K}_{rr})\det(\mathbf{K}_{tt} - \kappa^2\mathbf{K}_{rr}^{-1}) = 0 \,,$$

the characteristic equation in κ readily following

$$P(\kappa) \equiv \det(\mathbf{K}_{tt} - \kappa^2\mathbf{K}_{rr}^{-1}) = 0 \,, \tag{16.45}$$

which is apparently a cubic equation in κ^2, the outcome being that the eigenvalues of the *decoupled* 6×6 Cartesian stiffness matrix occur in symmetric pairs, i. e., the six eigenscrews of the matrix occur in pairs of opposite chirality, while the absolute values of each pair are identical [16.62, 63].

16.5.2 Mobile Robots

Mobile robots bear an architecture similar to PKMs. Different from the latter, their limbs are not attached to a base plate, but free to move. In fact, their links proximal to the ground end by either feet or wheels, thereby leading to *weheeled* or, correspondingly, *legged* mobile robots. Mobile robots termed *hybrid* have been designed with a combination of wheeled and legged limbs. These robots take on various morphologies, as outlined below.

Wheeled Robots

The earliest of these robots were termed *autonomous guided vehicles* (AGV). They were provided with a chassis and two motors, to implement the 2-DOF proper of any terrestrial vehicle equipped with conventional – as opposed to omnidirectional – wheels. These systems were capable of following a trajectory printed on the flat ground of a manufacturing facility. They had thus a rather limited mobility, but their load-carrying capacity was virtually unlimited.

Robots with conventional wheels are generally equipped with two pairs of pneumatic tires, that allow them to adapt themselves to irregular surfaces. Moreover, these robots can either bear steerable wheels, like MDA's *Rover Chassis Prototype (RCP)*, or change course by means of maneuvers involving different angular velocities of their tires mounted on axles fixed to the chassis, like the *Khepera* robot.

Extraterrestrial exploration has prompted the development of *rovers*, namely, wheeled robots intended for roaming on unstructured terrain, either soft, like the Moon's or Mars's regolith, or hard, like rocks. Design issues here pertain mainly to the wheels, as these must adapt themselves to a rich variety of conditions and terrain constitution. Rovers are currently designed to travel at very low speeds, sometimes of a few meters per day, the case of NASA's *Curiosity* (This rover landed on Mars on August 6, 2012.). The main issue here is for the wheels to provide enough traction and prevent spinning without advancing. The latter is guaranteed by the use of individual motors on each wheel. Traction is enhanced by means of *grousers* on the wheel periphery. *Curiosity* and *RCP* feature metal wheels with grousers.

Legged Robots

This kind of robots has evolved tremendously since the earlier systems developed in the eighties: *Ohio state university (OSU) hexapod* and the *OSU adaptive suspension vehicle (ASV)*, among others. Current legged robots range from those with six legs, *X-RHEX Lite*, with a planar symmetry and three identical legs on each side of the body, or the *multi-appendage robotic system (MARS)*, with its legs distributed along the vertices of a regular hexagon, to four-legged robots with an architecture similar to that of mammals, like *Big Dog* and *Cheetah*. Special mention deserve bipeds, of which the best known are humanoids, with the architecture of the human body, namely, *ASIMO* (advanced step in innovative mobility) and its competitor, *QRIO* (quest for curiosity). Design issues vary, depending on the number of legs. Hexapods offer a combination of static stability and maneuverability, which their four-legged counterparts do not. By far, humanoids are the highest demanding in terms of static and dynamic stability, but offer the highest mobility.

Hybrid Robots

Hybrid robots stem from the combination of the two foregoing types, wheeled with legged. These are systems that look like wheeled robots, but that feature wheels instead of feet. They offer the advantages – and the disadvantages – of the two foregoing types. A recent example of this kind of robots is NASA's ATHLETE (all-terrain hex-legged extra-terrestrial explorer) robot, that features six articulated legs stemming from the edges of a regular hexagon defining the robot body and ending in wheels. Its design is intended for service to the planned Lunar base. According to NASA, the 15-ton lunar habitat would be [16.67]

> *mounted on top of the six-legged robot. The habitat could walk right off of the lunar lander, and then proceed to any desired location. Wheeled locomotion would be used for level ground; more challenging terrain could be negotiated with the full use of the flexible legs.*

16.5.3 Aquatic Robots

Aquatic robots are currently being marketed for operation on water, whether on the surface or underneath. An example of a surface robot is the *Mantra*, intended to act as a lifeguard and help swimmers in distress. Some aquatic robots have been designed with the morphology of fish [16.68]. One swimming robot designed with a hexapod morphology, featuring six flippers in lieu of legs is *Aqua*, developed at McGill University, and depicted in Fig. 16.5.

By far, the aquatic robot with the highest profile is the robotic submarine *Bluefin*, which became a celebrity

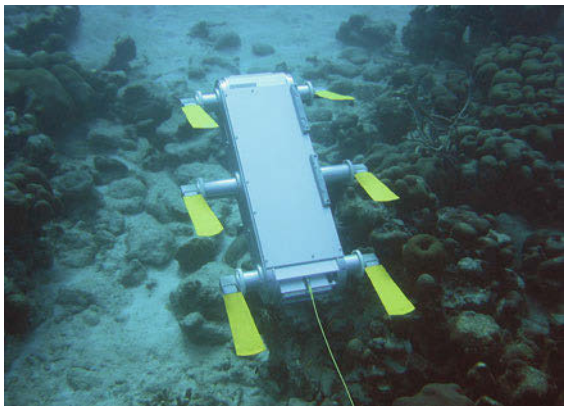


Fig. 16.5 Aqua, an amphibious robot

in the search for the debris of Flight MHS370 (Flight went missing on March 7, 2014 at 16:41 GMT.).

16.5.4 Flying Robots

This field is becoming quite active, with some robots mimicking the morphology of insects, and falling into the category of micro-robots [16.69]; other flying robots are designed as unmanned aerial vehicles (UAV) [16.70]. Intensive research is now being reported in the design and control of two novel types of UAVs, namely, drones and quadrotors. The control of drones undergoing fast maneuvers has been made possible by the inception of gyro-free inertial measurement systems fabricated with microelectromechanical systems (MEMS) technology. Highly maneuverable quadrotors, especially suitable for surveillance and reconnaissance missions, are small-size, light-weight flying machines supplied with *two pairs of counter-rotating rotors and propellers, located at the vertices of a square frame* [16.71]. A recent issue of IEEE's *Robotics & Automation Magazine* [16.72] includes a state-of-the-art account of quadrotor technology.

16.6 Summary

This chapter is devoted to the design of robots, with a focus on serial architectures. In this regard, we started by proposing a stepwise design procedure; then, we recalled the main issues in robot design. These issues pertain to workspace geometry, and to the kinetostatic, the dynamic, the elastostatic and the elastodynamic performances. In doing this, the mathematics behind the concepts addressed was briefly outlined, so as to make the chapter self-contained.

A major design issue in the design of PKM, namely, stiffness, was given due attention. Design issues pertaining to other kinds of robots, like wheeled, legged, flying and aquatic, were outlined.

References

16.1 O. Bottema, B. Roth: *Theoretical Kinematics* (North-Holland, Amsterdam 1979), also available by Dover Publishing, New York 1990

16.2 J. Angeles: The qualitative synthesis of parallel manipulators, ASME J. Mech. Des. **126**(4), 617–624 (2004)

16.3 R. Hartenberg, J. Denavit: *Kinematic Synthesis of Linkages* (McGraw-Hill, New York 1964)

16.4 G. Pahl, W. Beitz: *Engineering Design. A Systematic Approach*, 3rd edn. (Springer, London 2007), translated from the original Sixth Edition in German

16.5 M. Petterson, J. Andersson, P. Krus, X. Feng, D. Wäppling: Industrial robot design optimization in the conceptual design phase, Proc. IEEE Int. Conf. Mechatron. Robotics, Vol. 2, ed. by P. Drews (2004) pp. 125–130

16.6 R. Clavel: Device for the movement and positioning of an element in space, US Patent 497 6582 (1990)

16.7 R. Vijaykumar, K.J. Waldron, M.J. Tsai: Geometric optimization of serial chain manipulator structures for working volume and dexterity, Int. J. Robotics Res. **5**(2), 91–103 (1986)

16.8 J.P. Merlet: *Parallel Robots* (Kluwer, Boston 2006)

16.9 J. Angeles: *Fundamentals of Robotic Mechanical Systems*, 3rd edn. (Springer, New York 2007)

16.10 K. Hoffman: *Analysis in Euclidean Space* (Prentice Hall, Englewood Cliffs 1975)

16.11 L. Burmester: *Lehrbuch der Kinematik* (Arthur Felix, Leipzig 1886)

16.12 J.M. McCarthy: *Geometric Design of Linkages* (Springer, New York 2000)

16.13 J. Angeles: Is there a characteristic length of a rigid-body displacement?, Mech. Mach. Theory **41**, 884–896 (2006)

16.14 H. Li: Ein Verfahren zur vollständigen Lösung der Rückwärtstransformation für Industrieroboter mit allegemeiner Geometrie, Ph.D. Thesis (Universität-Gesamthochschule Duisburg, Duisburg 1990)

16.15 M. Raghavan, B. Roth: Kinematic analysis of the 6R manipulator of general geometry, Proc. 5th Int. Symp. Robotics Res., ed. by H. Miura, S. Arimoto (MIT, Cambridge 1990)

16.16 J. Angeles: The degree of freedom of parallel robots: A group-theoretic approach, Proc. IEEE Int. Conf. Robotics Autom. (ICRA), Barcelona (2005) pp. 1017–1024

16.17 C.C. Lee, J.M. Hervé: Translational parallel manipulators with doubly planar limbs, Mech. Mach. Theory **41**, 433–455 (2006)

16.18 B. Roth: Performance evaluation of manipulators from a kinematic viewpoint, National Bureau of Standards – NBS SP **495**, 39–62 (1976)

16.19 A. Kumar, K.J. Waldron: The workspaces of a mechanical manipulator, ASME J. Mech. Des. **103**, 665–672 (1981)

16.20 D.C.H. Yang, T.W. Lee: On the workspace of mechanical manipulators, ASME J. Mech. Trans. Autom. Des. **105**, 62–69 (1983)

16.21 Y.C. Tsai, A.H. Soni: An algorithm for the workspace of a general n-R robot, ASME J. Mech. Trans. Autom. Des. **105**, 52–57 (1985)

16.22 K.C. Gupta, B. Roth: Design considerations for manipulator workspace, ASME J. Mech. Des. **104**, 704–711 (1982)

16.23 K.C. Gupta: On the nature of robot workspace, Int. J. Robotics Res. **5**(2), 112–121 (1986)

16.24 F. Freudenstein, E. Primrose: On the analysis and synthesis of the workspace of a three-link, turning-pair connected robot arm, ASME J. Mech. Trans. Autom. Des. **106**, 365–370 (1984)

16.25 C.C. Lin, F. Freudenstein: Optimization of the workspace of a three-link turning-pair connected robot arm, Int. J. Robotics Res. **5**(2), 91–103 (1986)

16.26 T. Yoshikawa: Manipulability of robotic mechanisms, Int. J. Robotics Res. **4**(2), 3–9 (1985)

16.27 J. Loncaric: Geometric Analysis of Compliant Mechanisms in Robotics, Ph.D. Thesis (Harvard University, Cambridge 1985)

16.28 B. Paden, S. Sastry: Optimal kinematic design of 6R manipulators, Int. J. Robotics Res. **7**(2), 43–61 (1988)

16.29 J.K. Salisbury, J.J. Craig: Articulated hands: Force control and kinematic issues, Int. J. Robotics Res. **1**(1), 4–17 (1982)

16.30 G.H. Golub, C.F. Van Loan: *Matrix Computations* (Johns Hopkins Univ. Press, Baltimore 1989)

16.31 G. Strang: *Linear Algebra and Its Applications*, 3rd edn. (Harcourt Brace Jovanovich, San Diego 1988)

16.32 A. Dubrulle: An optimum iteration for the matrix polar decomposition, Electron. Trans. Numer. Anal. **8**, 21–25 (1999)

16.33 W.A. Khan, J. Angeles: The kinetostatic optimization of robotic manipulators: The inverse and the direct problems, ASME J. Mech. Des. **128**, 168–178 (2006)

16.34 H. Pottmann, J. Wallner: *Computational Line Geometry* (Springer, Berlin, Heidelberg 2001)

16.35 H. Asada: A geometrical representation of manipulator dynamics and its application to arm design, Trans. ASME J. Dyn. Sys. Meas. Contr. **105**(3), 131–135 (1983)

16.36 T. Yoshikawa: Dynamic manipulability of robot manipulators, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (1985) pp. 1033–1038

16.37 P.A. Voglewede, I. Ebert-Uphoff: Measuring closeness to singularities for parallel manipulators, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (2004) pp. 4539–4544

16.38 A. Bowling, O. Khatib: The dynamic capability equations: A new tool for analyzing robotic manipulator performance, IEEE Trans. Robotics **21**(1), 115–123 (2005)

16.39 C.M. Gosselin, J. Angeles: A new performance index for the kinematic optimization of robotic manipulators, Proc. 20th ASME Mech. Conf. (1988) pp. 441–447

16.40 C. Gosselin, J. Angeles: The optimum kinematic design of a planar three-degree-of-freedom parallel manipulator, ASME J. Mech. Trans. Autom. Des. **110**, 35–41 (1988)

16.41 A. Bicchi, C. Melchiorri, D. Balluchi: On the mobility and manipulability of general multiple limb robots, IEEE Trans. Robotics Autom. **11**(2), 232–235 (1995)

16.42 P. Chiacchio, S. Chiaverini, L. Sciavicco, B. Siciliano: Global task space manipulability ellipsoids for multiple-arm systems, IEEE Trans. Robotics Autom. **7**, 678–685 (1991)

16.43 C. Melchiorri: Comments on Global task space manipulability ellipsoids for multiple-arm systems and further considerations, IEEE Trans. Robotics Autom. **9**, 232–235 (1993)

16.44 P. Chiacchio, S. Chiaverini, L. Sciavicco, B. Siciliano: Reply to comments on Global task space manipulability ellipsoids for multiple-arm systems' and further considerations, IEEE Trans. Robotics Autom. **9**, 235–236 (1993)

16.45 F.C. Park: Optimal robot design and differential geometry, ASME J. Mech. Des. **117**, 87–92 (1995)

16.46 F.C. Park, J. Kim: Manipulability of closed kinematic chains, ASME J. Mech. Des. **120**(4), 542–548 (1998)

16.47 A. Liégeois: Automatic supervisory control for the configuration and behavior of multibody mechanisms, IEEE Trans. Sys. Man. Cyber. **7**(12), 842–868 (1977)

16.48 C.A. Klein, C.H. Huang: Review of pseudo-inverse control for use with kinematically redundant manipulators, IEEE Trans. Syst. Man Cybern. **13**(2), 245–250 (1983)

16.49 J.M. Hollerbach: Optimum kinematic design of a seven degree of freedom manipulator. In: *Robotics Research: The Second International Symposium*, ed. by H. Hanafusa, H. Inoue (MIT, Cambridge 1985)

16.50 M.W. Spong: Remarks on robot dynamics: Canonical transformations and riemannian geometry, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (1992) pp. 454–472

16.51 T.J. Graettinger, B.H. Krogh: The acceleration radius: A global performance measure for robotic manipulators, IEEE J. Robotics Autom. **4**(11), 60–69 (1988)

16.52 M. Griffis, J. Duffy: Global stiffness modeling of a class of simple compliant couplings, Mech. Mach. Theory **28**, 207–224 (1993)

16.53 S. Howard, M. Zefran, V. Kumar: On the 6×6 cartesian stiffness matrix for three-dimensional motions, Mech. Mach. Theory **33**, 389–408 (1998)

16.54 L. Meirovitch: *Fundamentals of vibrations* (McGraw-Hill, Boston 2001)

16.55 C.M. Gosselin, J.-F. Hamel: The agile eye: A high-performance three-degree-of-freedom camera-orienting device, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (1994) pp. 781–786

16.56 O. Company, F. Pierrot, T. Shibukawa, K. Morita: Four-Degree-of-Freedom Parallel Robot, EU Patent No. EP 108 4802 (2001)

16.57 X. Kong, C. Gosselin: *Type synthesis of parallel mechanisms*, Springer Tract. Adv. Robotics, Vol. 33 (Springer, Berlin, Heidelberg 2007)

Avidbots Ex1016
Page 458 of 2154

16.58 J. Hervé: The Lie group of rigid body displacements, a fundamental tool for mechanism design, Mech. Mach. Theory **34**, 719–730 (1999)
16.59 W.A. Khan, J. Angeles: A novel paradigm for the qualitative synthesis of simple kinematic chains based on complexity measures, ASME J. Mech. Robotics **3**(3), 0310161–03101611 (2011)
16.60 J. Lončarivc: Normal forms of stiffness and compliance matrices, IEEE J. Robotics Autom. **3**(6), 567–572 (1987)
16.61 X. Ding, J.M. Selig: On the compliance of coiled springs, Int. J. Mech. Sci. **46**, 703–727 (2004)
16.62 J.M. Selig: The spatial stiffness matrix from simple stretched springs, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (2000) pp. 3314–3319
16.63 J. Angeles: On the nature of the Cartesian stiffness matrix, Rev. Soc. Mex. Ing. Mec. **3**(5), 163–170 (2010)
16.64 A.K. Pradeep, P.J. Yoder, R. Mukundan: On the use of dual-matrix exponentials in robotic kinematics, Int. J. Robotics Res. **8**(5), 57–66 (1989)
16.65 A. Taghvaeipour, J. Angeles, L. Lessard: On the elastostatic analysis of mechanical systems, Mech. Mach. Theory **58**, 202–216 (2013)
16.66 D. Zwillinger (Ed.): *CRC Standard Mathematical Tables and Formulae*, 31st edn. (CRC, Boca Raton 2002)
16.67 B. Christensen: Robotic lunar base with legs changes everything, http://www.space.com/5216-robotic-lunar-base-legs.html
16.68 L. Wen, T. Wang, G. Wu, J. Li: A novel method based on a force-feedback technique for the hydrodynamic investigation of kinematic effects on robotic fish, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (2011), Paper No. 0828
16.69 L.L. Hines, V. Arabagi, M. Sitti: Free-flight simulation and pitch-and-roll control experiments of a sub-gram flapping-flight micro aerial vehicle, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (2011), Paper No. 0602
16.70 C.E. Thorne, M. Yim: Towards the dvelopment of gyroscopically controled micro air vehicles, Proc. 2011 IEEE Int. Conf. Robotics Autom. (2011), Paper No. 1800
16.71 T. Lee, M. Leok, N.H. McClamroch: Geometric tracking control of a quadrotor UAV on SE(3), Proc. 49th IEEE Conf. Decis. Control (2010)
16.72 Robots take flight: Quadroto unmanned aerial vehicles, IEEE Robotics Autom. Mag. **19**(3) (2012), http://online.qmags.com/RAM0912?pg=3&mode=2#pg1&mode2

Avidbots Ex1016
Page 459 of 2154

Multimedia Contents

17. Limbed Systems

Shuuji Kajita, Christian Ott

A limbed system is a mobile robot with a body, legs and arms. First, its general design process is discussed in Sect. 17.1. Then we consider issues of conceptual design and observe designs of various existing robots in Sect. 17.2. As an example in detail, the design of a humanoid robot HRP-4C is shown in Sect. 17.3. To design a limbed system of good performance, it is important to take into account of actuation and control, like gravity compensation, limit cycle dynamics, template models, and backdrivable actuation. These are discussed in Sect. 17.4.

In Sect. 17.5, we overview divergence of limbed systems. We see odd legged walkers, leg–wheel hybrid robots, leg–arm hybrid robots, tethered walking robots, and wall-climbing robots. To compare limbed systems of different configurations, we can use performance indices such as the gait sensitivity norm, the Froude number, and the specific resistance, etc., which are introduced in Sect. 17.6.

17.1 **Design of Limbed Systems** 420

17.2 **Conceptual Design** 420
17.2.1 Conceptual Design Issues 421
17.2.2 Case Studies 421

17.3 **Whole Design Process Example** 423
17.3.1 Conceptual Design of HRP-4C 423
17.3.2 Mechanical Design 424
17.3.3 Electrical System Design 425

17.4 **Model Induced Design** 427
17.4.1 Dynamic Walking 427
17.4.2 Template Models 430
17.4.3 Robots for Control Based on Backdrivable Actuation 432

17.5 **Various Limbed Systems** 434
17.5.1 Odd Legged Walkers 434
17.5.2 Leg-Wheel Hybrid Robots 435
17.5.3 Leg-Arm Hybrid Robots 436
17.5.4 Tethered Walking Robots 436
17.5.5 Wall-Climbing Robots 436

17.6 **Performance Indices** 437
17.6.1 Stability Margins 437
17.6.2 Gait Sensitivity Norm 437
17.6.3 Duty Factor and Froude Number 438
17.6.4 Specific Resistance 439

Video-References 439

References 440

In this chapter, we refer a limbed system as a mobile robot which consists of a body, legs and arms. Its general concept is illustrated in Fig. 17.1.

Let us define that a limbed system must have a body and at least one leg (lower limb) to support and to propel itself. A leg interacts with the environment by its end effector (foot). It can also have an arbitrary number of arms (upper limb) to manipulate target objects by their end effectors (hands).

From a body-centered point of view, legs can be regarded as special arms designed for manipulating the earth. This justifies us to treat both of the legs and the arms as limbs in a unified manner, thus we chose the word *limbed systems* as the title of this chapter.

Avidbots Ex1016
Page 460 of 2154

17.1 Design of Limbed Systems

General robot design process has already been discussed in the previous 16, therefore, we will concentrate on the specific issues on limbed systems. A design process of actual limbed system can be seen in literatures [17.1–3]. A typical development process may takes the following steps:

- *Step 1*: *Basic requirements.* A designer must specify the class of tasks performed by the limbed system and its requirements, target velocity, payload, expected environment, etc.
- *Step 2*: *Conceptual design.* The number of legs and arms, their topology, and associated gait are decided simultaneously. They must be designed at the same time, because the leg structure and gait are tightly coupled, thus they should not be considered separately. Geometric dimensions of links are also determined based on the basic requirements.
- *Step 3*: *Detailed design.* A designer selects actuators, sensors, reduction gears, and other mechanical/electrical components to realize the conceptual design. Then the structural elements are designed to integrate the devices.
- *Step 4*: *Evaluation.* Evaluate the design by building a prototype model or performing computer simulations. In our experience, the first design can never meet the basic requirement, and we are forced to return to Step 3 to change the motor type or the reduction ratio. Sometimes, the simulation results suggest us reconsidering the conceptual design or even rethinking the basic requirements.

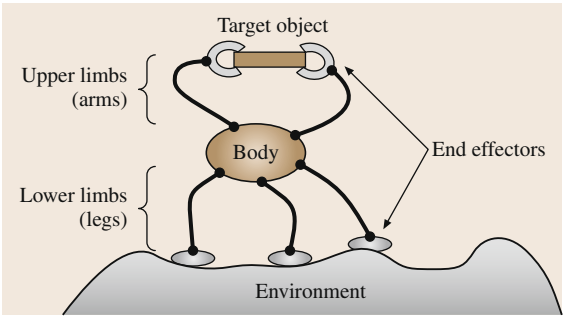


Fig. 17.1 General concept of a limbed system

Figure 17.2 overviews the development process of a limbed system. Note that a development of a limbed system is an extremely dynamic process, rather than a static top-down development. The detail of each step will be discussed in the following subsections.

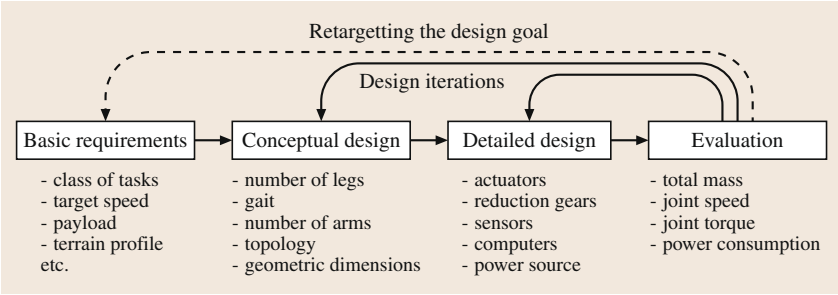


Fig. 17.2 Typical development process of a limbed system

17.2 Conceptual Design

At the beginning, a designer must have *a class of tasks* in his mind, for example:

- Enter a damaged nuclear power plant, climb stairs, and close specified valves
- Explore a planet to seek a new lifeforms
- Carry a person and climb stairs
- Perform dances to entertain its audience, etc..

Note that it must be a class of tasks and should not be a single task, because the latter can be realized by a single degree-of-freedom machinery as expressed in the former chapter.

A class of tasks should be compiled into more specific goals. For example, target speed of traveling, maximum step length, maximum payload, or expected terrain profiles (ex. maximum height of obstacles).

Those figures give a solid guide line for a design process.

17.2.1 Conceptual Design Issues

The goal of conceptual design is to decide the basic kinematic property of the robot, the number of legs and arms, limb topology, and geometric dimensions of the links. A robot designer should take into account the following issues:

- *Gait type*: Leg motion pattern for walking or running is called *gait* and it greatly affects limbed system design. For example, a robot for static gait should be designed to have sprawled posture to maximize its stability. Whereas, a robot for dynamic gait is designed to have erect posture with high center of mass to get a longer time constant of its falling. The gait generation is discussed in 75.
- *Biomimesis*: Some robots are designed to mimic the mechanical structure of a living creature as accurate as possible. This is beneficial for deeper understanding of biological systems. For example, *Quinn* and *Ritzmann* have built a hexapod robot with kinematics that are remarkably similar to the cockroach *Blaberus discoidalis* [17.4] (VIDEO 521). Biomimesis is also meaningful in terms of entertainment applications. *Hirukawa* et al. developed biped walking dinosaur robots of *Tyrannosaurus Rex* and *Parasaurolophus* for an exposition [17.5]. Those robots were built on 30% scale of the real dinosaurs due to safety issues; nevertheless, they could attract many audience by their realistic outlooks at the 2005 World Exposition, Aichi, Japan.
- *Bioinspired dynamics*: To reproduce the robust and versatile locomotion of animals, some designers pay their attention on essential dynamics of the locomotion, rather than the mechanical similarity. *Koditschek* et al. emphasized this standpoint on the development of their hexapod robot RHex, which equipped with only six actuators and yet can run on various irregular terrain [17.6].
- *Mechanical simplicity*: It is preferred to realize the target tasks with a mechanism as simple as possible. Therefore, the design is always biased to use fewer number of actuators. VIDEO 520 shows an example of robot design with minimum numbers of actuators.
- *Limb workspace*: A limb must have at least three degrees of freedom (DOF) to freely locate the tip of a limb in 3-D space. It must have also at least six DOF for the arbitrary orientation of the end effector in 3-D space.
- *Load bearing*: Proper joint assignment can reduce the joint torque to support the body weight. In VIDEO 517, we can see a unique hexapod leg design for efficient load bearing.

17.2.2 Case Studies

Let us observe how abovementioned issues are reflected in the conceptual design of actual limbed systems.

Underactuatedness Versus Redundancy

We compare two monopods which were designed under different policies on actuation. Figure 17.3a is the 3-D one-legged hopping machine developed by *Raibert* [17.1]. This robot is driven by one pneumatic cylinder in its leg and two hydraulic actuators in its hip joints. Therefore, it has only three actuated DOF while 3-D rigid body motion requires 6. Despite its underactuatedness, it can freely travel in 3-D space keeping its body upright thanks to its balance control.

Figure 17.3b shows the one-legged robot developed by *Tajima* and *Suga* [17.7]. It has seven actuated DOFs in total, three at the hip, one at the knee, two at the ankle, and one at the toe. Therefore, this robot is a redundant system having one extra DOF in addition to the six required for 3-D motion. This redundancy can help the robot to reduce the excessive joint speed required at hopping. As a result, successful hopping motion control was realized by using servo motors with reduction gears for all joints.

Serial Versus Parallel

Legs of a limbed system can be configured either as a serial or parallel structures. In most cases, each leg of a biped robot is designed as serial chain of six ac-

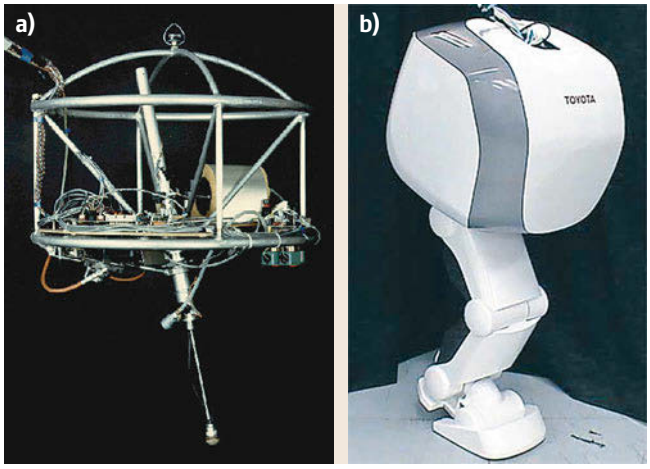


Fig.17.3a,b Monopod robot designs. **(a)** MIT 3-D One-Leg Hopper (1983); **(b)** Toyota monopod (2006)

tuated DOF, where three are used in the hip, one in the knee and two in the ankle. HRP-2 L developed by *Kaneko* et al. [17.8], as shown in Fig. 17.4a, is one of such examples. For such robots, the most difficult part is the hip joint design. A hip joint mechanism must contain three axis of rotation (roll/pitch/yaw) bearing high torque caused by the upper body weight. This is serious especially for a robot to carry heavy loads.

Takanishi et al. solved this problem by introducing legs of the Stewart platform for their WL-16R, a human carrying biped locomotor (Fig. 17.4b). Each leg consists of six linear actuators driven by DC-geared motors and ball screws. In this design, their robot can carry a human of 50 kg weight [17.9].

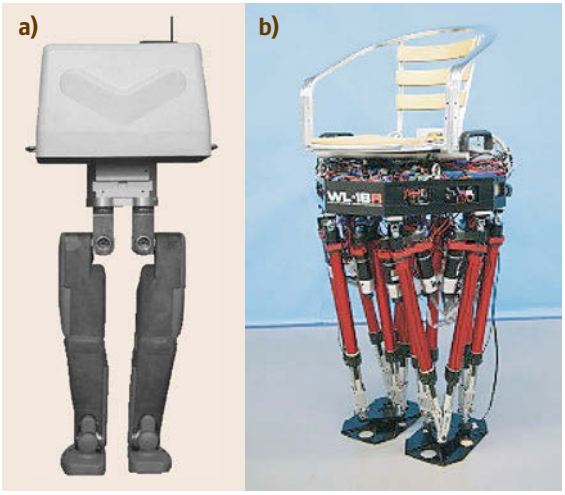


Fig.17.4a,b Biped robot designs. **(a)** HRP-2L (2001); **(b)** WL-16R (2003)

Static Gait Versus Dynamic Gait

Figure 17.5a is TITAN III developed by Hirose and his colleagues [17.10]. This is one of the successful quadruped robots in 1980s. Each leg is a 3-D pantograph mechanism driven (PANTOMEC) by three linear actuators (ball screw with DC motor) equipped on its body, thus it has 12 DOF. To perform statically balanced walk, this robot was designed to have a low center of mass and widely spread legs.

Figure 17.5b is BigDog, an energy autonomous hydraulic quadruped robot of 1 m tall, 1 m long, and 90 kg developed by *Buehler* et al. [17.11]. Each leg has one passive linear pneumatic compliance in the lower leg, three active joints for knee, hip pitch, and roll. There-

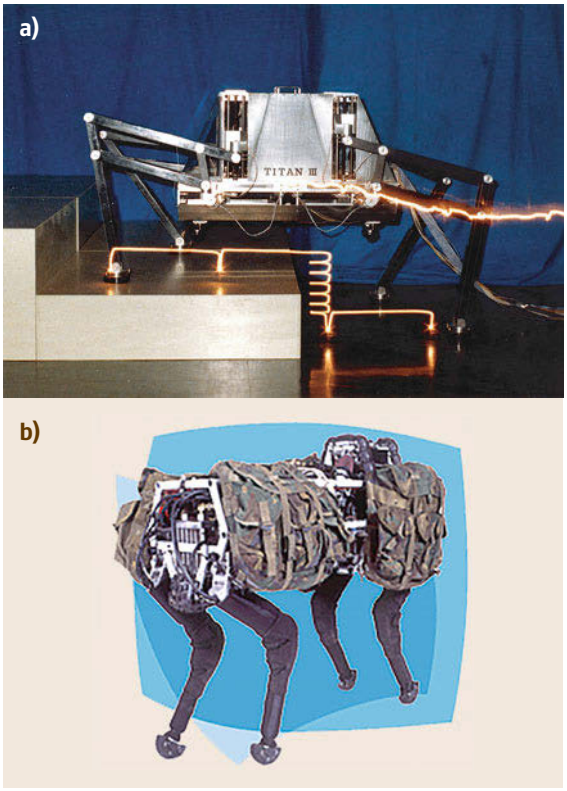


Fig.17.5a,b Quadruped designs. **(a)** TITAN III (1984); **(b)** BigDog (2005)

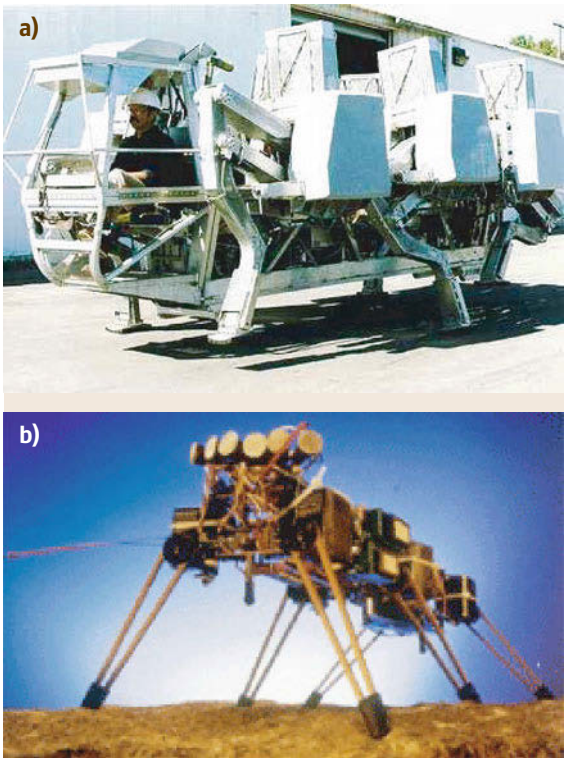


Fig.17.6a,b Hexapod designs. **(a)** ASV (1986); **(b)** Genghis (1989)

Avidbots Ex1016
Page 463 of 2154

fore it has 12 actuated DOF and four passive DOF. This robot can demonstrate remarkable walking performance in an outdoor environment, as well as can maintain its balance even when kicked by a human. This ability of dynamic balancing is helped by its design of high center of mass with the leg assignment with narrow lateral separation.

Big Versus Small

Design concept is deeply affected by the robot size. Figure 17.6a shows one of the most famous hexapods, the adaptive suspension vehicle (ASV) developed by *Waldron* et al. [17.2, 12]. The ASV is a hydraulically driven hexapod robot which can carry a parson over rough terrain. Its length and height are 5.2 m and 3.0 m, respectively, and weighs 2700 kg. To walk around in 3-D space, each leg requires 3 DOFs, thus the robot has 18 DOF in total.

Figure 17.6b shows a small hexapod robot Genghis (35 cm length and 1 kg weight) developed by *Brooks* [17.13]. This robot could also demonstrate robust walking in 3-D space, but its weight is less than a thousandth of the ASV. As a result, Genghis does not need precise 3-D foot positioning for its walking control. Each leg has only two DOF and the robot has 12 DOF in total. Standing with three legs, the body position and orientation can be fully controlled as long as the feet is allowed to slip on the ground.

Degree of Biomimesis

We can increase the degree of biomimesis by adding DOF; however, fewer DOF is preferable in terms of engineering. The first version of ASIMO has 26 DOF in total, 6 for each leg, 5 for each arm, one for each hand, and two for the head (Fig. 17.7a) [17.14]. As a biped walking humanoid robot, this is a reasonable configuration.

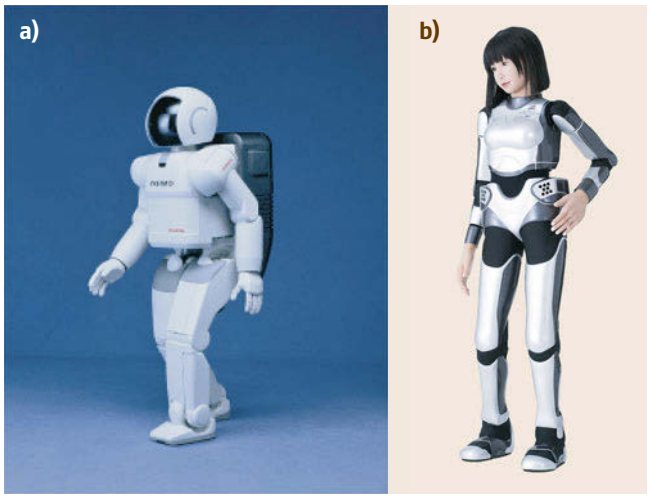


Fig.17.7a,b Humanoid designs. **(a)** ASIMO (2000); **(b)** HRP-4C (2009)

Cybernetic human HRP-4C (Fig. 17.7b) was designed to be as close as a human in consideration of applications in entertainment industry [17.15, 16]. It has 44 DOF in total, 7 for each leg, 6 for each arm, 2 for each hand and, three for its waist, 3 for its neck, and 8 for its face. Its design detail is discussed in the next section.

17.3 Whole Design Process Example

In this section, we explain the design process of a humanoid robot HRP-4C. We do this to give the readers an example of the whole development process of a limbed system. The readers are also recommended to see the comprehensive development report for the humanoid robot LOLA written by *Lohmeier* [17.3]. Also please watch another successful humanoid designs in VIDEO 522 and VIDEO 526. Note that a general mechanical design and construction process have already been discussed in this handbook (Chap. 4).

17.3.1 Conceptual Design of HRP-4C

At the beginning of our project, we defined *Cybernetic human* as a humanoid robot with the following features:

1. Have the appearance and shape of a human being.
2. Can walk and move like a human being.
3. Can interact with humans using speech recognition and so forth.

Such robots can be used in the entertainment industry, for example, exhibitions and fashion shows. It can also be used as a human simulator to evaluate devices for humans.

As the successor of our previous humanoid robots HRP-2 and HRP-3 [17.17, 18], we call our new humanoid robot HRP-4C, C stands for cybernetic human.

To determine the target shape and dimensions of HRP-4C, we used the anthropometric database for Japanese population, which was measured and compiled by *Kouchi* et al. [17.19]. The database provides

the dimensions of different Japanese groups of ages and sex. Considering the entertainment applications like fashion show, we picked the average young female data (age from 19 to 29). Figure 17.8 shows part of the dimensions provided by the database.

To reproduce women's graceful motion, we asked a professional walking model to perform walking, turning, sitting on chair, and other motions. The performance was motion-captured and used to evaluate possible joint configurations proposed for HRP-4C structure. From the captured data, the required joint workspace was estimated. In addition, we estimated the motor power during biped walking to design the appropriate leg joint configuration.

Figure 17.9 shows the joint configuration we finally decided for HRP-4C. To reproduce the human motion, three DOFs (roll, pitch, yaw) were assigned for each of the waist and the neck joints. In addition, we put active toe joints to realize biped walking with wide stride.

17.3.2 Mechanical Design

Figure 17.10 shows the mechanism to drive the knees of HRP-4C. The rotation of the servo motor is transmitted to the input of the harmonic drive gear by way of the pulleys and the timing belt, and the output of the harmonic gear is connected to the lower leg link. By this way, we can obtain high torque output without backlash which is required for dynamic biped walking. Note that

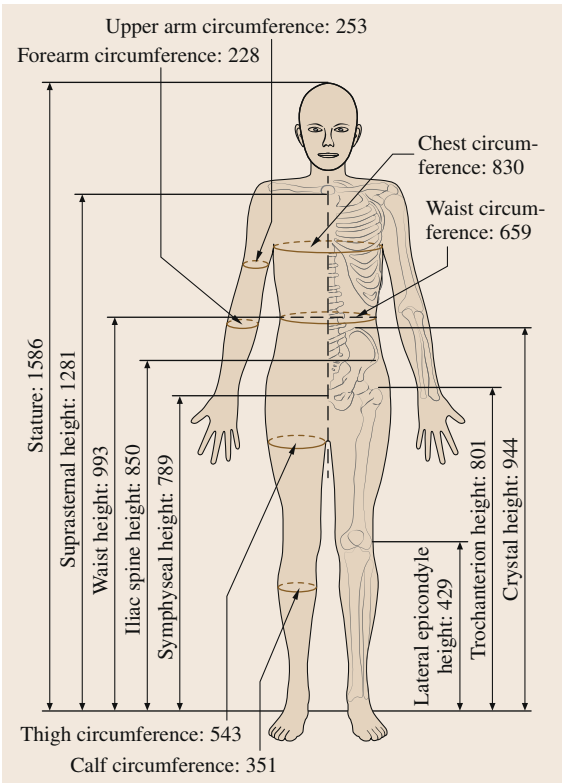


Fig. 17.8 Anthropometric data of average young Japanese female (after [17.19], all measures in mm)

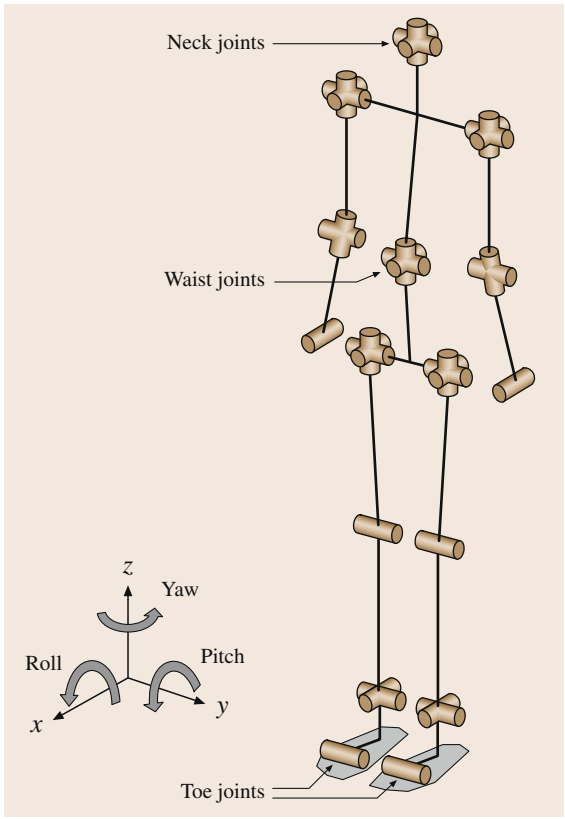


Fig. 17.9 Joint configuration of HRP-4C body (joints for the face and hands are omitted)

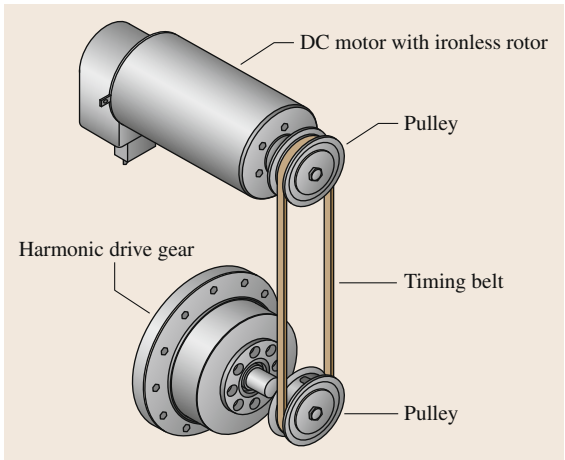


Fig. 17.10 Joint drive system (after [17.8])

the pulleys and the timing belt offer us a flexible choice of the motor placement as well as the total reduction ratio. The belt compliance becomes negligible thanks to the high reduction ratio of the harmonic gear at the final stage.

The same mechanisms of Fig. 17.10 are used for most of the joints of HRP-4C; however, there are some exceptions. The first example is the ankle joint. Figure 17.11 illustrates the mechanism of the left lower leg link and the ankle. The ankle pitch and roll joints are driven by two servomotors (Motor #1 and Motor #2) embedded in the shin link. Motor #1 rotates the ball screw to create the linear motion via timing belt and pulleys. Then the ball screw's nut pushes or pulls the connecting rod that finally drives the ankle pitch joint.

Motor #2's rotation is transmitted to the bevel gear #1, timing belt #2, and the second bevel gear. The bevel gear #2 rotates the harmonic drive gear and it finally rotates the ankle roll joint. This elegant yet complicated mechanism was necessary to realize the slim outer shape of the HRP-4C ankle which must have close dimensions of human ankle.

Figure 17.12 is the toe joint mechanism of HRP-4C. To avoid sole plate separation which may catch obstacles between the plates, the toe joint axis should locate as close as possible to the floor surface. For this purpose, a four bar linkage is used to drive the toe sole plate by the harmonic drive gear mounted on the heel sole plate.

As already mentioned, harmonic drive gears are mainly used because they are backlash free. On the other hand, for the mechanical parts, which do not require backlash-less motion, we can use alternative devices. For this reason, servo motors with planetary gear heads were used as the actuators for facial expressions and hands.

Figure 17.13 is the final mechanical design of HRP-4C (Fig. 17.13a) and its exterior (Fig. 17.13b). The covers for the arms, legs and body are made of fiber reinforced plastics (FRP) and the facial skin and hand skin are made of silicone rubber.

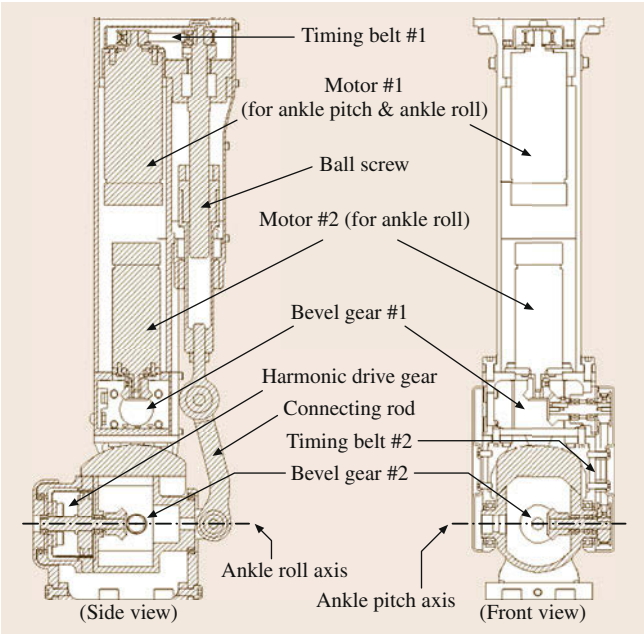


Fig. 17.11 Ankle joint drive mechanism of HRP-4C

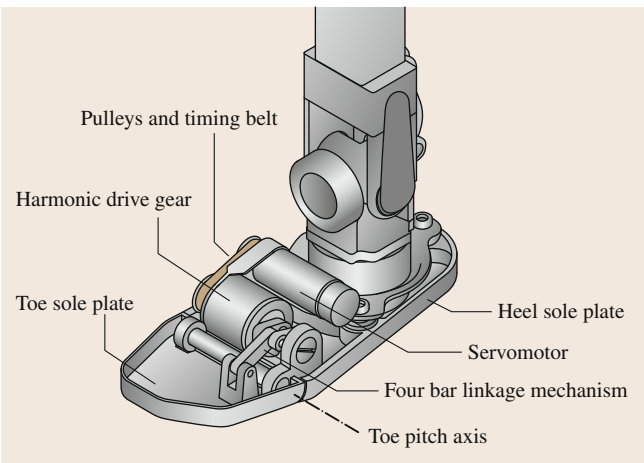


Fig. 17.12 Toe joint mechanism

17.3.3 Electrical System Design

Figure 17.14 shows the electronics architecture of HRP-4C. The entire system is controlled by a single board computer of the PCI-104 form factor with Intel Pentium M 1.6 GHz. Two peripheral boards, the 10 ch controller area network CAN interface board and the force sensor interface board are stacked on the CPU board.

To realize a human-like slender outlook, we used compact motor drivers which were developed in our previous project [17.18]. This device is an intelligent driver for a single servo motor, which takes an incremental encoder signal and provides motor current by proportional–integral–derivative (PID) control. The driver is small enough to be embedded in leg/arm links nearby the target motor. Since it has a CAN interface which allows cascade connection from the CAN interface board, the wiring can be simplified and reduced.

In addition, we used another CAN-interfaced intelligent drivers for small motors for the face and the hands. This driver can drive multiple motors by handling each incremental encoder, PID feedback, and the current control. We developed this type of

Avidbots Ex1016
Page 466 of 2154

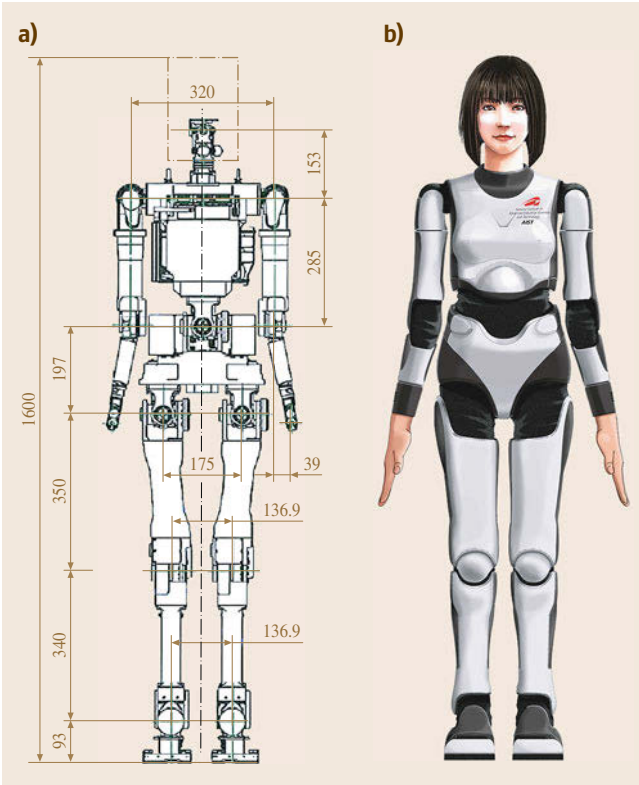


Fig.17.13a,b Mechanism of (a) HRP-4C and its (b) exterior

Table 17.1 Principal specifications of HRP-4C

Height		1,600 mm
Weight		48 kg (with batteries)
Total DOF		44 DOF
	Face	8 DOF
	Neck	3 DOF
	Arm	6 DOF × 2
	Hand	2 DOF × 2
	Waist	3 DOF
	Leg	7 DOF × 2
CPU		Intel Pentium M 1.6 GHz
Sensors	Head	CCD camera
	Body	Inertial measurement unit (IMU)
	Sole	6-axis force sensor × 2
Batteries		NiMH 48 V

driver by modifying the device for our multifingered hand [17.20].

By using these motor drivers and the CAN interface board, we can control 44 servo motors in total. Furthermore, a commercial inertial measurement unit (IMU) is connected via CAN to measure and control of the body posture during biped walk.

The feet of HRP-4C are equipped with six-axis force sensors to measure the zero-moment point (ZMP) for its balance control. These sensor signals are interfaced by the force sensor interface board.

Table 17.1 shows the principal dimensions and specification of the final design of HRP-4C. It becomes

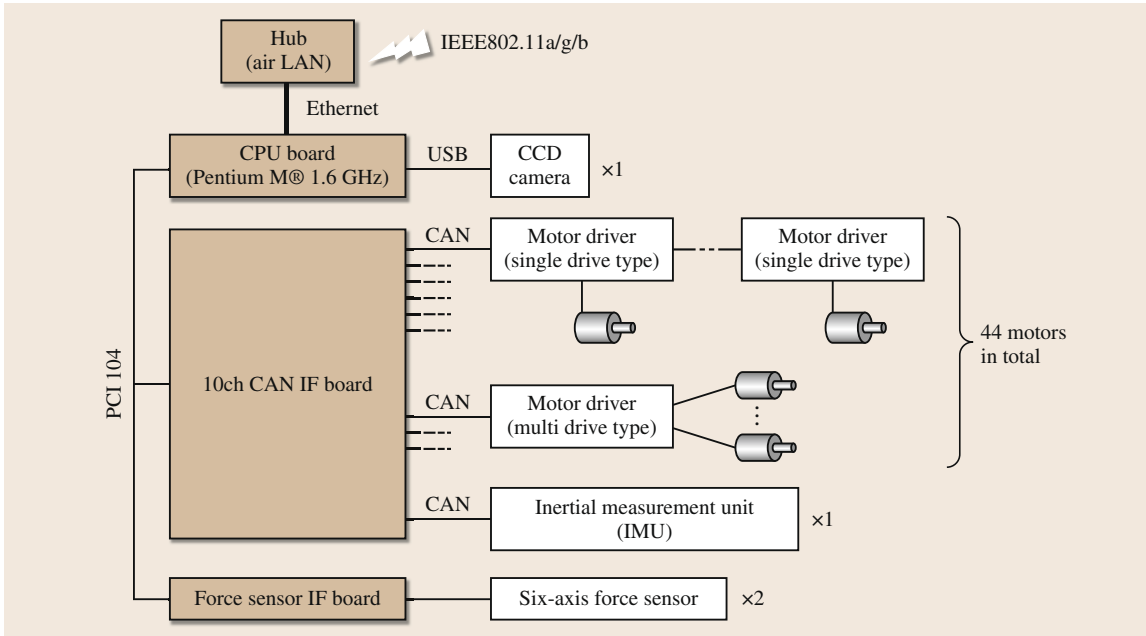


Fig. 17.14 Electronics architecture of HRP-4C

a humanoid robot with 1.6 m height, 46 kg weight, and 44 actuated DOF. The on-board nickel metal hydride battery (NiMH) provides power for 20 min operation time without an external cable. HRP-4C's human-like walking demonstration and a quick slip-turn experiment are shown in VIDEO 522 and VIDEO 525, respectively.

17.4 Model Induced Design

The design of multilimbed robotic systems is a mechatronic challenge involving expertise in diverse fields such as mechanical design, electrical design, communication and control, as well as software engineering. A task-oriented perspective on the overall system allows in some cases to incorporate a desired function directly into the system design rather than enforcing this behavior by active control. This idea can be found in many robotic designs. The most prominent example might be the use of counterweights for passive gravity compensation employed in robot manipulators. A clever mechanical design thus leads to a smaller power consumption of the actuators and consequently leads to a simplified system design. Passive gravity balancing has been realized via several approaches involving counterweights, springs (Fig. 17.15), and parallel mechanisms [17.21]. Such a gravity compensation based on springs is for example employed in the robot PR2 from Willow Garage (Fig. 17.16) [17.22, 23]. In this section, we will highlight several examples of legged robots where a desired function or property of the final system is realized at least partly via clever mechanical design.

17.4.1 Dynamic Walking

Passive Dynamic Walking

The concept of passive dynamic walking was pioneered by McGeer (VIDEO 527). In the seminal paper [17.24], a class of purely passive mechanical systems on an inclined surface were analyzed which exhibit a periodic gait that is intrinsically, i. e., open loop, stable. The resulting periodic motion is characterized by the fact that the increase of energy due to the slope is exactly balanced by the energy loss at the impact. McGeer proposed to analyze these systems in terms of orbital stability using Poincaré maps.

A detailed analysis of simple walking models with point feet, which are based on this principle, can be found for instance in [17.25] or [17.26]. Figure 17.17 shows a conceptual model of a passive walking machine which is able to walk down on an inclined slope. For analyzing the dynamics of this system, the following two components are needed: 1) The open-loop dynamics of the swing phase: Under the assumption that the stance leg is standing firmly on the ground (without lift-off or slipping) until the next touch-down, the open loop dynamics can be derived from standard Lagrangian multibody dynamics. 2) An impact model

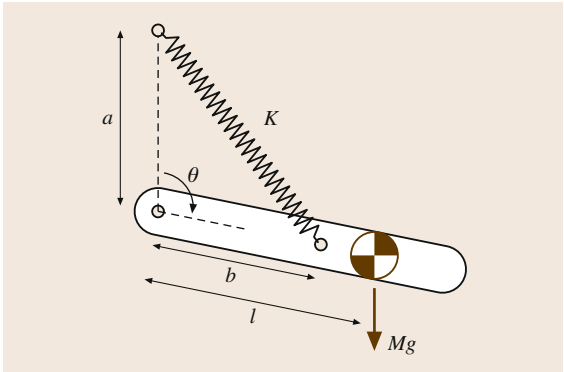


Fig. 17.15 Passive gravity compensation by a spring: For $Kab = Mgl$ the torques around the hinge produced by gravity and by the spring counter-balance each other



Fig. 17.16 PR2 robot from Willow Garage using springs for passive gravity compensation (after [17.22, 23])

Avidbots Ex1016
Page 468 of 2154

at the time of touch-down of the swing leg, which determines the state variables after the impact based on the state variables before the impact. Often, a perfectly inelastic impact is assumed.

In a fully passive mechanism with given dynamics parameters, the motion is completely determined by the initial condition of the system. The mapping from the initial state to the state at the beginning of the next step is called *stride function* or *return map*. A periodic gait can be characterized by a fixed point of the stride function, and the search for a periodic gait can accordingly be formulated as an optimization problem on the stride function. In general, optimization may involve the initial condition as well as the dynamics parameters including the inclination of the slope.

In order to analyze the stability properties of a periodic gait, McGeer proposed to interpret the stride function as a Poincaré map of the system. Local (orbital) stability can then be decided based on the linearization of the Poincaré map. Since computing the Poincaré map (i. e., computing the stride function) involves solving the open loop dynamics of the system until the next touch-down, this often can only be implemented numerically.

Figure 17.18 shows technical realizations of planar passive dynamic walkers. The walker shown in Fig. 17.18a from Technical University of Delft has the simplest possible configuration with straight legs and point-feet. Premature impact of the swing leg with the ground is avoided by a stepping pattern on the inclined slope. The mechanism in Fig. 17.18b was built at Cornell University based on ideas from McGeer's seminal paper. The arc shape of the foot implements a rolling motion during stance. The ground clearance of the swing leg was realized by a passive knee joint.

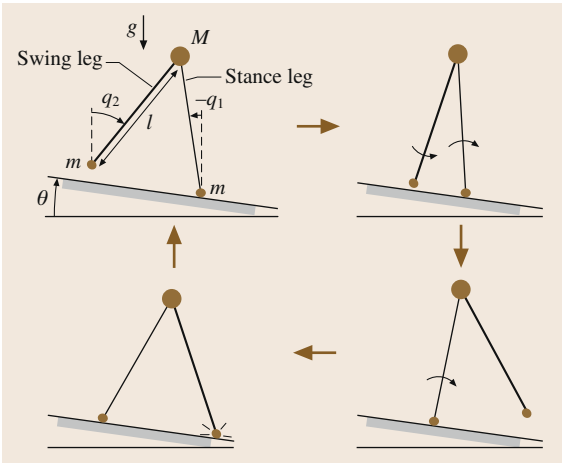


Fig. 17.17 Conceptual model of a simple passive dynamic walker

Figure 17.19 shows a fully passive walking machine in 3-D [17.27] (VIDEO 532). It employs soft heels for improving the contact transition. The swinging motion of the arms allows for reducing the angular momentum around the vertical axis as well as side-to-side rocking. Knee extension is limited to a straight configuration by a locking mechanism.

Semipassive Limit Cycle Walker

Purely passive dynamic walkers can produce a stable limit cycle when walking down a shallow slope. Thereby, the energy loss at each impact is compensated

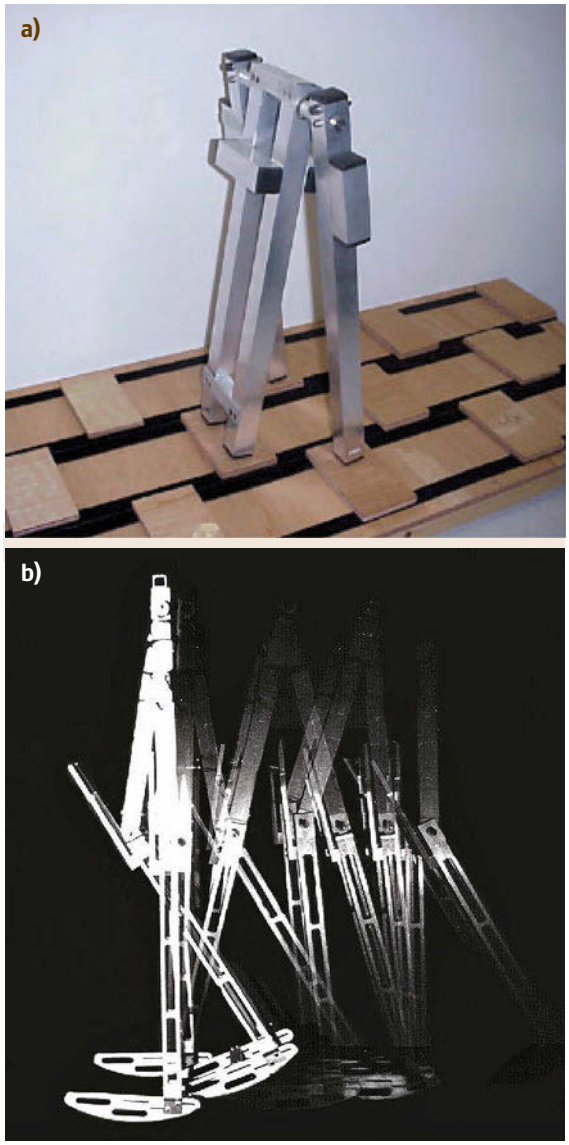


Fig.17.18a,b Examples of planar passive dynamic walkers with **(a)** point feet, **(b)** arc feet, and knees

by the energy gain from the decent along the slope, i. e., these systems are *powered* by gravity.

Several attempts have been made for increasing the versatility and robustness of limit cycle walkers by adding active power sources. Figure 17.20 shows some successful designs reported in [17.28]. In these systems, simple actuation is used for substituting gravitational power in order to allow for walking on level ground. While these systems are not fully passive, their energy consumption is much less than compared to fully actuated walking robots. The Cornell biped, as shown in Fig. 17.20a, utilizes actuation only in the ankle. The actuation is used such that each ankle joint extends when the opposite foot hits the ground. The Delft biped, as shown in Fig. 17.20b, instead is powered at the hip and utilizes passive ankles.

The superior energy efficiency which can be achieved by dynamic limit cycle walkers has been demonstrated in the Cornell Ranger [17.29] that succeeded to walk over a distance of 65 km without recharging or direct human intervention except for the remotely controlled steering.



Fig. 17.19 A 3-D passive walker with upper body (after [17.27])

Actuated Limit Cycle Walkers

Passive and semipassive limit cycle walkers strongly rely on the passive dynamics for realizing a specific gait. In actuated limit cycle walkers the control of some joints is used for imposing a limit cycle on the remaining dynamics. The systems shown in Fig. 17.21 are fully actuated, except for some unactuated degrees of freedom, e.g., at the foot–ground contact, and/or underactuation due to elasticity.

In the framework of the *hybrid zero dynamics* [17.32], a cyclic variable like the virtual leg angle (angle from the hip to the stance foot) is introduced and the active degrees of freedom are controlled via partial feedback linearization to implement a virtual constraint which describes how the active degrees of freedom are coupled to the cyclic variable. The shape of the virtual constraint is then optimized aiming at a stable limit cycle for the remaining two-dimensional (2-D) zero dynamics. A comprehensive treatment of this approach can be found in [17.33]. Such control approaches were implemented on the robots RABBIT [17.34] and MABLE [17.35]. While RABBIT employed position control in the active degrees of freedom, MABLE was implemented with series elastic actuators allowing for torque-based control (VIDEO 533).

The design of the 3-D walking machine Flame in Fig. 17.21c was also motivated by the aim of combining limit cycle walking with active control [17.36]. The actuation of this robot was also based on series elastic

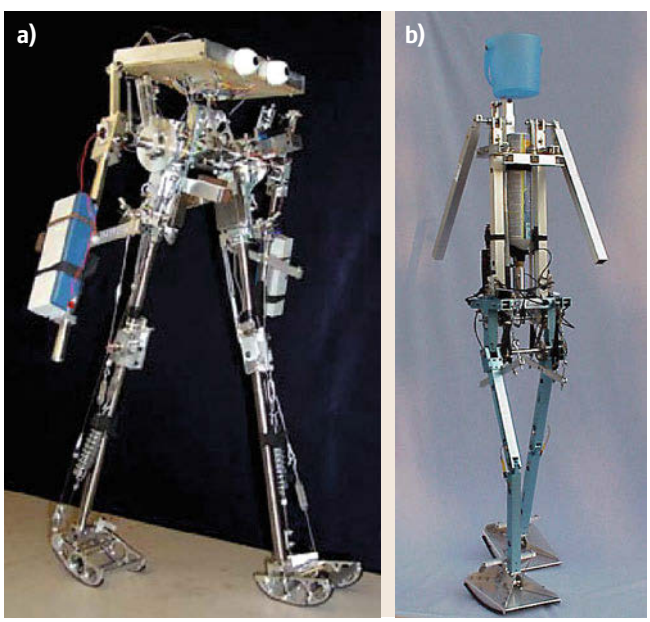


Fig.17.20a,b Underactuated biped robots: **(a)** 3-D passive walker (Cornell Biped, 2003) developed at Cornell Univ. (after [17.30]); **(b)** Denise (2004) (after [17.31])

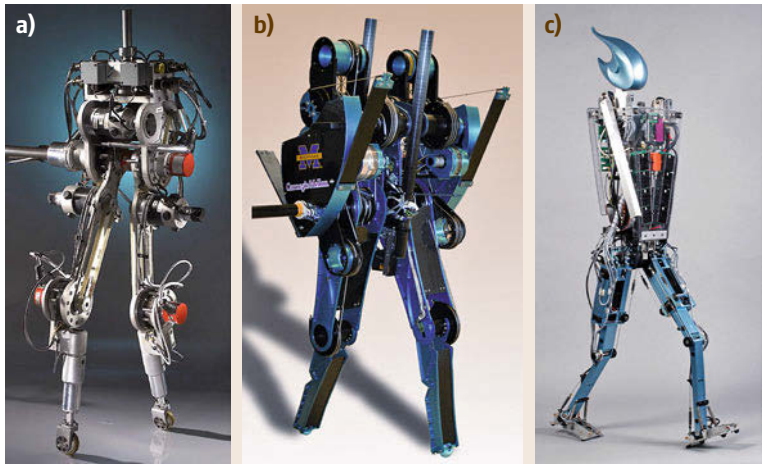


Fig.17.21a–c Actuated limit cycle walkers **(a)** RABBIT (2002); **(b)** MABEL (2008); **(c)** Flame (2007)

actuation and it was controlled by an event-based state machine in combination with active foot placement for gait stabilization.

17.4.2 Template Models

Conceptual models play an important role as templates for the design and control of legged walking machines. Template models can also be used for generating hypotheses about the neuro-mechanical control in biological systems [17.37]. In this context, they provide a possible answer to Bernstein's DOF problem, how the redundancy lying in the human neuro-muscular system is coordinated. We will discuss two conceptual models which have proven useful for legged locomotion.

The linear inverted pendulum (LIP) model has been proposed as a template model for bipedal walking (VIDEO 512). Figure 17.22a shows the main forces acting on the center of mass (COM) of a legged robot. If all the mass is concentrated in one single point, which implies mass-less legs, the acceleration forces of the COM are counter balanced by the ground interaction force F acting at the ZMP p. If the COM is kept at a constant height, i. e., $\ddot{z} = 0$, one immediately obtains the dynamical equations of the linear inverted pendulum model

$$\ddot{x} = \frac{g}{z}(x - p) \,. \tag{17.1}$$

This model has been the starting point for many successful methods aiming at trajectory generation and feedback control of bipedal humanoid robots [17.41–43]. More details on the related control issues can be found in 48. By using this model, the implicit assumption is made that the change of angular momentum around the COM, e.g., due to the swing leg dynamics,

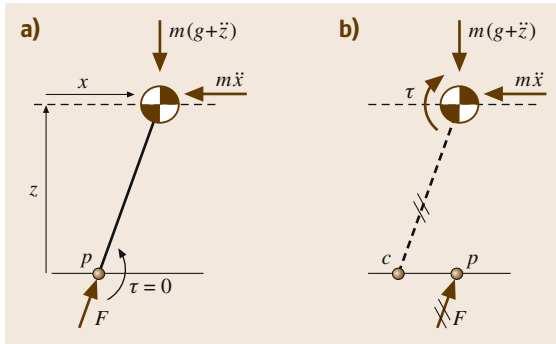


Fig. 17.22 **(a)** The linear inverted pendulum model (after [17.38]) **(b)** Definition of the centroidal moment pivot (after [17.39, 40])

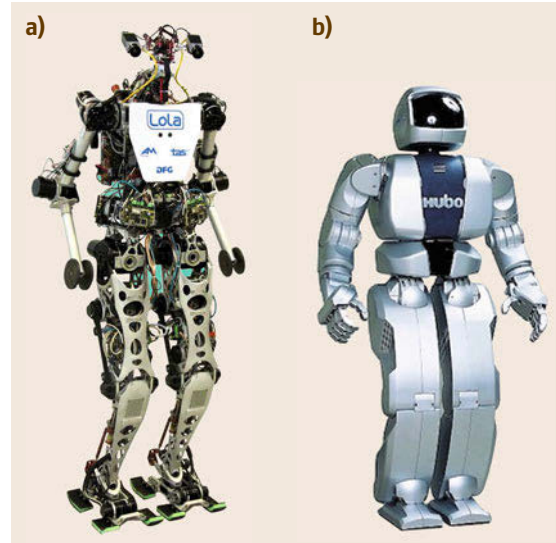


Fig.17.23a,b Bipedal robots **(a)** LOLA (2009); **(b)** HUBO (2005)

can be neglected. This assumption plays an important role also for the mechanical design. The bipedal robot LOLA (Fig. 17.23a) developed at Technical University of Munich (TUM) is optimized for low inertia in the legs by moving the knee and ankle actuators into thigh [17.44]. In the robot HUBO from KAIST (Fig. 17.23b) the design aimed at concentrating the main inertia into the torso while keeping the mass of the moving parts in the arms and legs small [17.45].

Aiming at a better representation of the change of angular momentum the centroidal moment pivot (CMP) (Fig. 17.22b) was proposed as an additional ground reference point [17.39, 40]. For the more general model in Fig. 17.22b the force balance results in

$$\ddot{x} = \frac{g+\ddot{z}}{z}(x-p) + \frac{\tau}{mz} \,. \tag{17.2}$$

The CMP is defined as the hypothetical location of the ZMP for the case $\tau = 0$, such that the distance between the ZMP and the CMP is proportional to the change of angular momentum, i. e.,

$$c = p - \frac{\tau}{F_z} \,, \tag{17.3}$$

where $F_z = m(g+\ddot{z})$. In undisturbed human walking it was observed that the distance between ZMP and CMP is kept small [17.40], while the change of angular momentum plays a stronger role for push recovery [17.46].

An alternative way of modeling the effect of the limb motion (e.g., swing foot dynamics) on the overall dynamics is to consider a three-mass model, which has been utilized in the control of LOLA [17.44] and the Honda humanoid robot ASIMO [17.47].

While the LIP model has been introduced as a conceptual model for walking, it cannot describe running motions in which a flight phase is present. In biomechanics, the spring-loaded inverted pendulum (SLIP) model (Fig. 17.24) was introduced in which the leg is modeled as a linear spring [17.48].

During the stance phase, the dynamics is given by

$$m\ddot{\boldsymbol{r}} = m\boldsymbol{g} + k\left(\frac{l_0}{|\boldsymbol{r}-\boldsymbol{r}_\mathrm{F}|} - 1\right)(\boldsymbol{r}-\boldsymbol{r}_\mathrm{F}) \,, \tag{17.4}$$

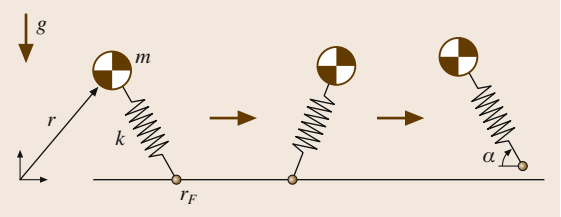


Fig. 17.24 The spring loaded inverted pendulum (SLIP) model

where l_0 and k are the rest length and stiffness of the spring, respectively, and $\boldsymbol{r}_\mathrm{F}$ is the location of the stance foot. During the flight phase, one simply has $\ddot{\boldsymbol{r}} = \boldsymbol{g}$, and it is assumed that the leg angle instantaneously adjusts to an angle α for preparing the next touch-down. The model contains no damping and impact phenomena are eliminated by assuming mass-less legs. Therefore,

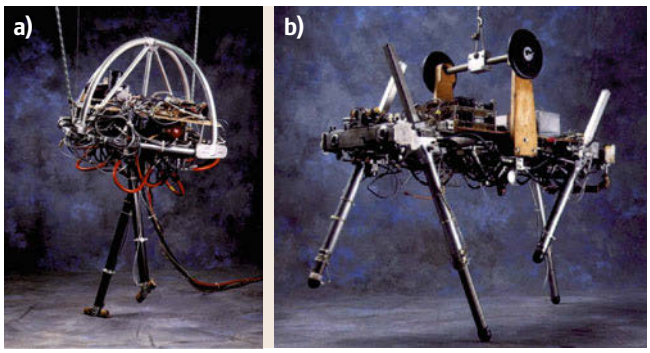


Fig.17.25a,b Hopping robots developed in the MIT LegLab **(a)** 3-D Biped (1989); **(b)** Quadruped (1984) (after [17.1])

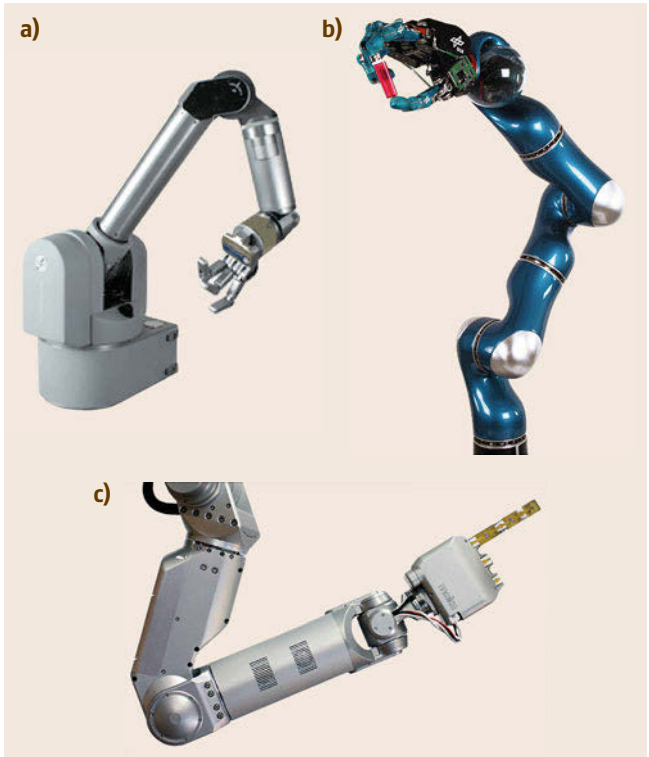


Fig.17.26a–c Robotic arms for implementing torque control: **(a)** WAM arm (Barrett Technologies) utilizing direct drive actuation in combination with cable transmissions, **(b)** the DLR lightweight-robot-III with integrated joint torque sensors, **(c)** the robot arm A2 (Mekka Robotics) based on series elastic actuators

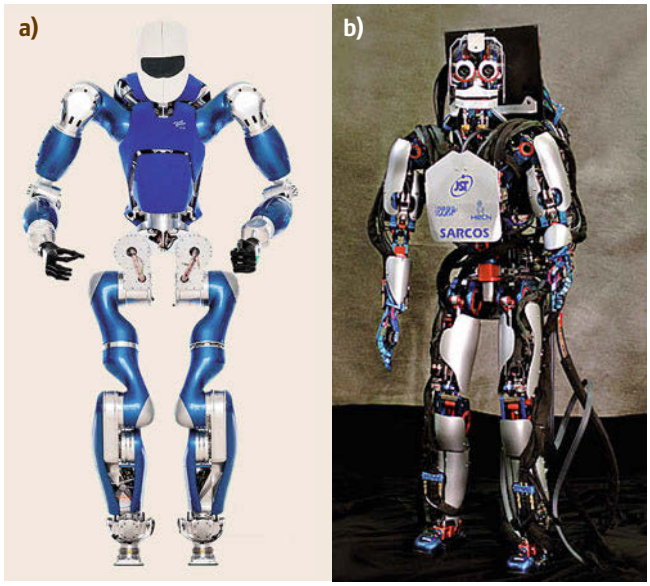


Fig.17.27a,b Fully torque controlled humanoid robots: **(a)** DLRs robot TORO (2013); **(b)** Humanoid robot CB based on hydraulic actuation (CB-i, 2008)

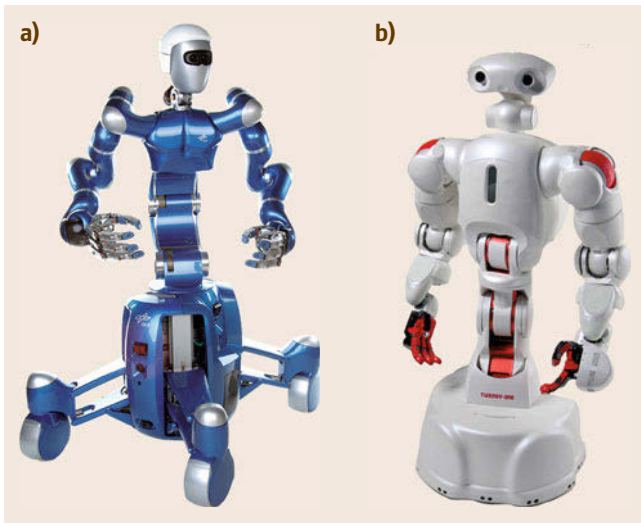


Fig.17.28a,b Humanoid upper body robots which allow whole body control based on joint torque: **(a)** DLRs torque controlled robot Justin and **(b)** Twendy-one from Waseda Univ. using SEA

this model is conservative. It allows to generate periodic hopping trajectories representing running motions. Moreover, the contact force profile resulting from these motions shows similarity to data from human running motions. An extension of the SLIP dynamics to bipedal walking including double support phases was analyzed in [17.49].

The SLIP model can be considered as the simplest model of one-legged hopping and bipedal running. Its compliant leg function can be of interest for the design and control of future legged robots, which aim at highly dynamic motions like fast running, hopping, and jumping.

Gait generation and control of compliant hopping robots have been studied in the 1980s by Raibert in the MIT LegLab. A detailed treatment of these developments is summarized in [17.1]. The basic idea of Raibert's dynamic locomotion approach can be explained by considering a planar one-legged hopping robot. It can be shown that this system can be controlled in a surprisingly simple way by the combination of three separate components:

1. Altitude control which is implemented by a fixed thrust during stance.
2. Control of the forward velocity by placing the foot at a fixed distance from the hip at touch-down.
3. Control of the body attitude during stance using actuation at the hip.

The same principle was applied to a one-legged hopper in 3-D, as well as to bipedal and quadrupedal hopping robots (Fig. 17.25). In the quadrupedal case, the combined effect of alternating pairs of legs was summarized by considering a single virtual leg.

17.4.3 Robots for Control Based on Backdrivable Actuation

A large number of algorithms for model-based control of robotic systems assume a rigid-body model of the robot in which the joint torques act as the control input. The capability of controlling the torque in practice is heavily influenced by the properties of the actuator transmission. While direct drive actuation allows a precise feed-forward control of the actuator torque via the motor current, this leads to rather heavy drive units. The WAM arm from Barrett Technologies (Fig. 17.26a) utilizes direct drive actuation in combination with cable transmissions in order to store the leading four drives into the base of the robot. In this way, a large inertia of the moving elements is avoided. The use of gears with high transmission ratios allows to reduce the overall weight, but leads to slower drive units which are hardly or not at all backdrivable. Joint torque sensing and serial elastic actuation present two hardware-oriented approaches for improving the torque controllability of highly geared actuators.

Joint torque sensing assumes a dedicated sensor for measuring the link side torque τ and was applied in a series of lightweight robot arms de-

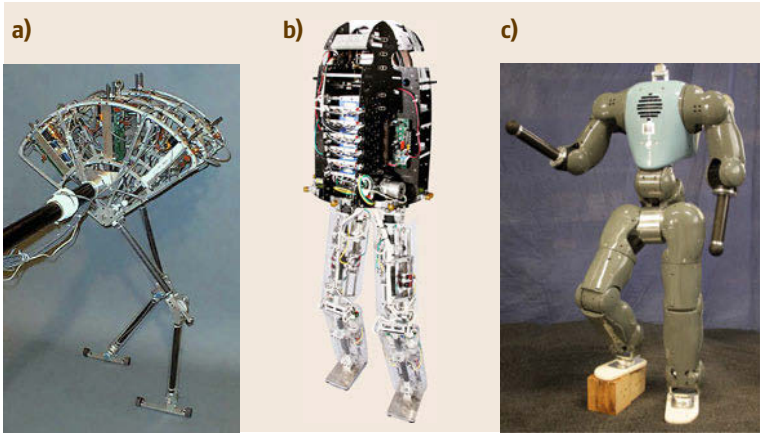


Fig.17.29a–c Bipedal robots using serial elastic actuators: **(a)** Spring Flamingo (1996) (after [17.50]); **(b)** M2V2 (2008) (after [17.51]); **(c)** COMAN (2011) developed at IIT utilizing SEA in selected joints (hip, knee, ankle)

veloped at DLR, including the light-weight robot in Fig. 17.26b [17.52] and the upper body robot Justin as shown in Fig. 17.28a [17.53]. Recently, the same drive technology has been integrated in the design of the torque-controlled humanoid robot TORO (Fig. 17.27a) [17.54] (VIDEO 531). Torque sensing and control have also been employed in the hydraulic humanoid CB developed by Sarcos (Fig. 17.27b).

In series elastic actuators (SEAs), the joint torque sensing is implemented indirectly by measuring the deflection of an elastic element which is introduced into the drive train [17.55]. Early robots with compliant actuators were the two armed systems COG [17.56] from MIT and Wendy [17.57] from Waseda university, which even allowed the adaptation of the compliance. More recently, series elastic actuators are used in the compliant arm A2 from Mekka (Fig. 17.26c) and the humanoid upper body robot Twendy-one from Waseda university (Fig. 17.28). They have been also applied to bipedal systems in the robots as shown in Fig. 17.29 (also watch VIDEO 529 and VIDEO 530).

While the mathematical models of torque controlled robots and robots with SEA have the same structure it should be mentioned that the source of elasticity is different in these approaches. In SEA, the deflective element for the torque sensing represents the main source of elasticity and its role is to decouple the motor and link side dynamics for high frequency disturbances like impacts. A well-designed torque sensor instead should not introduce a big effect on the overall joint stiffness (in this case usually the gear introduces the main source of elasticity). Despite their conceptual similarity, the literature of SEA and elastic joint robots also differs in the way that in the context of SEA usually the linear dynamics of a single compliant actuator is considered for the controller design, while in the context of elastic joint robots the full nonlinear multibody dynamics is considered.

From a modeling and control point of view, the use of torque sensors in the joints leads to a robot model with elastic joints. For actuators with a high transmission ratio, a common modeling assumption is that the kinetic energy of each rotor depends only on its own spinning motion and not on the rigid body motion of the other joints [17.58]. Let the motor and link side joint angles be denoted by $\theta \in \mathbb{R}^n$ and $q \in \mathbb{R}^n$, then the *reduced flexible joint model* (Chap 11 for a detailed derivation) is given by

$$\mathbf{M}(q)\ddot{q} + \mathbf{C}(q,\dot{q})\dot{q} + \boldsymbol{g}(q) = \boldsymbol{\tau} + \tau_{\mathrm{f},q}\,, \tag{17.5}$$

$$\mathbf{B}\ddot{\theta} + \boldsymbol{\tau} = \tau_{\mathrm{m}} + \tau_{\mathrm{f},\theta}\,, \tag{17.6}$$

where $\mathbf{M}(q)$, $\mathbf{C}(q,\dot{q})\dot{q}$, and $\boldsymbol{g}(q)$ represent the link side inertia matrix, the centrifugal and Coriolis forces, and the gravity term of the rigid body dynamics. The motor inertia is given by the diagonal matrix $\mathbf{B}$. The vector $\boldsymbol{\tau}$ represents the joint torques. In the case of an elastic joint with linear joint stiffness k_i the torque is given by $\tau_i = k_i(\theta_i - q_i)$. The terms $\tau_{\mathrm{f},q}$ and $\tau_{\mathrm{f},\theta}$ represent friction terms at the link side and the motor side, respectively.

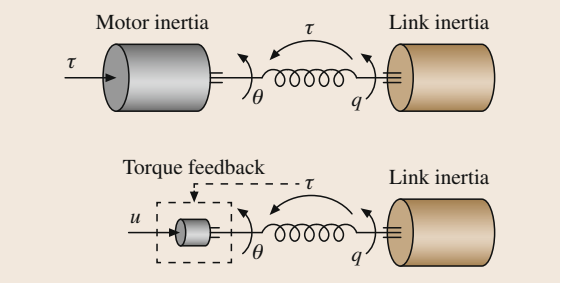


Fig. 17.30 Conceptual model of an elastic joint: A proportional feedback of the joint torque reduces the effective motor inertia and motor friction

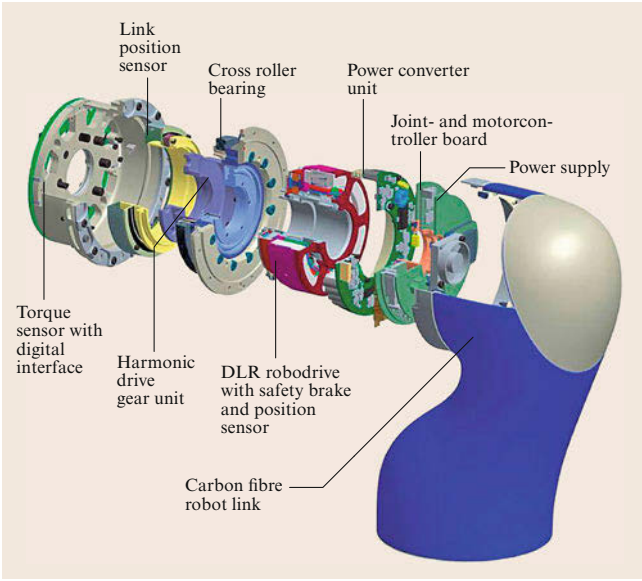


Fig. 17.31 The torque sensing based joint technology of DLR's light weight robot arm

A purely proportional feedback of the joint torques

$$\tau_{\mathrm{m}} = -(\mathbf{K}_\tau - I)\tau + \mathbf{K}_\tau u \tag{17.7}$$

has the effect that the motor dynamics equation (17.6) shows a controlled behavior as

$$\mathbf{K}_\tau^{-1}\mathbf{B}\ddot{\theta} + \tau = u + \mathbf{K}_\tau^{-1}\tau_{\mathrm{f},\theta}\ . \tag{17.8}$$

Let us assume that $\mathbf{K}_\tau$ is chosen as a diagonal matrix. Then we see that the matrix $\mathbf{K}_\tau^{-1}\mathbf{B}$ corresponds to an effective (i. e., virtual) motor inertia matrix. By choosing the gains in $\mathbf{K}_\tau$ larger than unity, we can thus reduce the effective motor inertia (Fig. 17.30). Moreover also the effects of motor friction are reduced such that a torque controlled joint ultimately exhibits a backdrivable behavior even if the underlying gear mechanism is not or only hardly backdrivable.

From this interpretation of torque feedback, we can derive some general guidelines for the mechanical design of torque-controlled robots. Since the effective motor side friction can be reduced by a simple proportional torque feedback but link side friction is not affected, we conclude that the mechanical design should aim at low friction at the link side and the torque sensing should be implemented at the power output side of the gear. Since the assembly of gear and torque sensor is assumed to behave like an ideal spring, any masses of moving parts in the transmission should be kept low in order to minimize unmodeled dynamics which would affect the validity of (17.7). As an example of a successful design, Fig. 17.31 shows a cross-section of a joint of the torque-controlled DLR-light-weight-robot-III [17.52] in which a harmonic drive transmission is combined with a strain gauge-based torque sensor.

Instead of the simple proportional control of (17.7), a complete torque controller can be designed based on the torque dynamics

$$\mathbf{B}\mathbf{K}^{-1}\ddot{\tau} + \tau = \tau_{\mathrm{m}} + \tau_{\mathrm{f},\theta} + \mathbf{B}\ddot{q}\ , \tag{17.9}$$

resulting from (17.6) and $\tau = K(\theta - q)$. Torque tracking is achieved via

$$\tau_{\mathrm{m}} = \tau_{\mathrm{d}} + \mathbf{B}\mathbf{K}^{-1}\ddot{\tau}_{\mathrm{d}} + \tau_{\mathrm{f},\theta} + K_q\mathbf{B}\ddot{q} + u(\tau_{\mathrm{d}} - \tau)\ , \tag{17.10}$$

where $u(\tau_{\mathrm{d}} - \tau)$ represents a proportional–derivative (PD) or PID control law and the nonnegative factor $K_q < 1$ is chosen less than 1 in order to avoid an overcompensation of the link side acceleration. In the context of SEA such a controller was proposed based on a linear system representation of a single joint in [17.55]. Cascaded torque control of a single SEA-based joint using an inner loop velocity controller has been analyzed in [17.59, 60], also using a linear model. In the context of torque-controlled elastic joint robots, in [17.61] a combination with a Cartesian impedance controller was proven to be stable for the full nonlinear dynamics based on the stability theory of cascaded systems (using $K_q = 1$). Passivity-based control has been analyzed in [17.62].

17.5 Various Limbed Systems

In this section, we explore the divergence of the limbed system design.

17.5.1 Odd Legged Walkers

A robot can be designed by pure engineering. For example, the tripodal walking robots STriDER (self-excited tripodal dynamic experimental robot) 1 and 2 (Fig. 17.32) can demonstrate a unique locomotion which have never developed by evolution [17.63] and VIDEO 534. It has three identical legs each of them has four DOF, three at the hip and one at the knee. Supporting the body with two legs, the robot takes a dynamic step by swinging the rest leg. Since the body flips

Avidbots Ex1016
Page 475 of 2154



Fig. 17.32 STriDER 1 and 2

with 180 ∘ about the aligned pelvis joints connecting the support legs, the robot can continue its dynamic walk.

Five legged walkers have been developed by some groups. For example, *Besari* et al. developed a five-legged robot inspired from starfish, and applied reinforcement learning to obtain its optimal gait [17.64].

17.5.2 Leg–Wheel Hybrid Robots

Unlike animals or insects, a robot can be designed to have wheels which can rotate infinitely. By mixing the efficiency of wheels and the flexibility of legs, we can expect a robot of maximum terrain adaptivity with minimum power consumption. Figure 17.33 shows examples of such a design concept.

Figure 17.33a shows a stair climbing of a biped leg-wheeled robot developed by *Matsumoto* et al. [17.65]. The robot is a planer biped with telescopic legs, but the tip of each leg is equipped with a powered wheel. During a single leg support phase, the robot is controlled as a wheeled inverted pendulum. In addition, a controller was developed to realize the smooth transient between single support and statically stable double support phase.

Figure 17.33b is RollerWalker developed by *Hirose* et al. [17.66, 69]. RollerWalker is a 12 DOF quadruped robot equipped with passive wheels on the tip of legs. It uses roller-skating mode on a flat floor, while it can walk on an uneven terrain by retracting the passive wheels (VIDEO 535).

Figure 17.33c is RHex developed by Buehler, *Koditchek* et al. [17.67]. Although it was originally inspired from the locomotion of cockroach, RHex has only six active DOF, that is, one actuator for each hip. Moreover, the legs can rotate full circle around the pitch axis. By this unique design, RHex can walk and run over rugged, broken, and obstacle-ridden ground (VIDEO 536). Recently, it also demonstrated biped running with its rear legs [17.70].

Figure 17.33d is Whegs II, another cockroach-inspired robot developed by *Allen* et al. [17.68]. This robot has only 4 active DOFs, one for propulsion, two for staring and one for body flexion. Each leg is equipped with three spring-loaded spoke and driven by

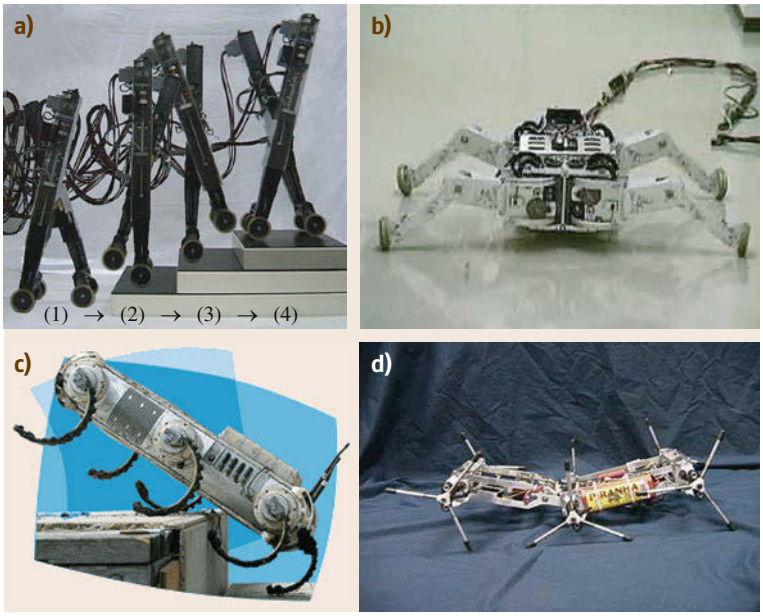
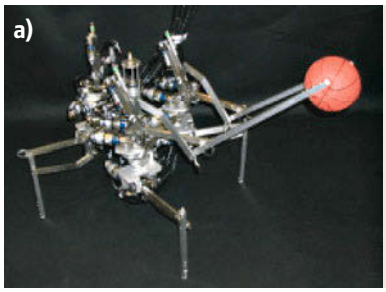


Fig.17.33a–d Leg-wheel hybrid robots. **(a)** Biped leg–wheel robot (1998); **(b)** Roller walker (1996); **(c)** RHex (2001); **(d)** Whegs II (2003) (after [17.65–68])



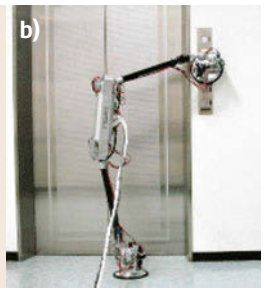


Fig.17.34a,b Leg-arm hybrid robots **(a)** MELMANTIS-1 (1996); **(b)** Yanbo3 (2003) (after [17.72, 73]) ◄

the same actuator (VIDEO 537). Whegs II can realize comparable mobility to RHex, while it uses fewer actuators.

17.5.3 Leg–Arm Hybrid Robots

Another design concept is a leg–arm hybrid robot. Since legs have inherently many degree of freedom, it is possible to use them as manipulators. By this way, we can minimize total DOF, complexity, weight and power consumption of the walking robot. MELMANTIS-1 (Fig. 17.34a) developed by *Koyachi* et al. is a hexapod walker of 22 DOF, which can transform its legs into manipulators [17.72]. The robot can manipulate an object by two legs while standing with four other legs, after traveling by six legs with the maximum stability.

Yanbo3 is a biped walker of 8 DOF developed by the group of *Ota* et al. [17.73]. It is designed to have minimum DOF necessary for a biped robot as well as for a manipulator when it is in the single support. In Fig. 17.34b, the robot is pressing the elevator button by its foot.

17.5.4 Tethered Walking Robots

Figure 17.35 shows Dante II, an eight-legged tethered walking robot developed by CMU Field Robotics Center in 1994. It was used at an Alaskan volcano for scientific exploration. To descend down steep crater walls in a rappelling-like manner, the robot uses tether cable anchored at the crater rim [17.71]. *Hirose* et al. also developed tethered quadruped for construction work [17.74].



Fig. 17.35 Dante II (1994) (after [17.71])

17.5.5 Wall-Climbing Robots

Wall-climbing robots are characterized by their foot mechanisms and leg configurations. The vital part is the foot mechanism to generate pulling force and the use of vacuum suction cups, electromagnets (for steel wall), adhesive materials, or miniature spine array has been proposed.

Figure 17.36a shows a wall-climbing quadruped NINJA-1 developed by *Hirose* et al. [17.75] Each foot of NINJA-1 is equipped with a specially designed suction pad which can minimize its vacuum leakage. Another reliable wall-climbing robot with suction cups was developed by *Yano* et al. [17.76]

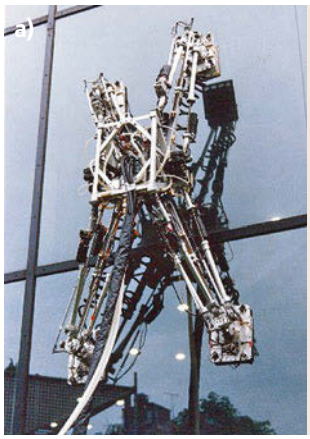




Fig.17.36a,b Wall-climbing robots. **(a)** NINJA-1 (1991); **(b)** RiSE (2005) (after [17.75, 77]) ◄

Figure 17.36b shows a wall-climbing hexapod RiSE developed by *Kim* et al. [17.77, 78]. Each foot of RiSE is equipped with arrays of miniature spines observed in some insects and spiders. The robot can reliably climb on a wide variety of outdoor surfaces including concrete, stucco, brick, and dressed sandstone. As a wall-climbing robot more recently developed, watch Stickybot III in VIDEO 540 and Waalbot in VIDEO 541.

17.6 Performance Indices

In this section, we introduce useful performance indices, that can be used to evaluate legged robots of different configurations.

17.6.1 Stability Margins

The stability margin was originally proposed for a degree of stability of the statically walking multilegged robots by *McGhee* and *Frank* [17.79]. By neglecting the inertial effects caused by body and leg acceleration, we can guarantee the robot keep balance if the projection of center of mass (COM) exists inside the support polygon as shown in Fig. 17.37. (Note: In multilegged robot research, the word *support pattern* is frequently used for support polygon [17.2, 79, 80].)

For a given configuration of a walking robot, the *stability margin* S_m is defined as the minimum distance of the vertical projection of COM to the boundaries of the support pattern in the horizontal plane as illustrated in Fig. 17.38a.

In addition, an alternative index was proposed to obtain the optimal gait analytically. That is the *longitudinal stability margin* S_l which is defined as the minimum distance from the vertical projection of COM to the support pattern boundaries along the line parallel to the body motion.

For a dynamic walking robot, we can define a stability margin as the minimum distance of the ZMP to the boundaries of the support polygon, since ZMP is the natural extension of a projected COM on the ground.

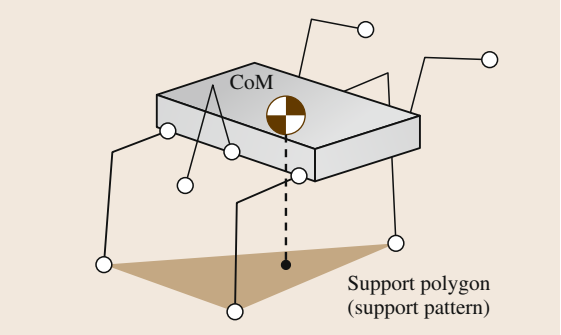


Fig. 17.37 Support polygon (support pattern) of multilegged robot

This fact has already been mentioned in the original work of ZMP [17.81], and has been implicitly used by many researchers. Explicit definition of this *ZMP stability margin* can be seen, for example, in *Huang* et al. [17.82].

For a legged robot on a rough terrain, *Messuri* and *Klein* defined the *energy stability margin* as the minimum potential energy required to tumble the robot as,

$$S_E = \min_i (Mgh_i) \,, \tag{17.11}$$

where h_i is the COM height variation during the tumble around the i-th segment of the support polygon and M is the total mass of the robot [17.83]. This concept is widely accepted, and there are some proposals for improvement [17.14, 84].

17.6.2 Gait Sensitivity Norm

Hobbelen and *Wisse* proposed a gait sensitivity norm (GSN) as a disturbance rejection measure of limit cycle walking machines [17.85]. GSN is defined as

$$\mathrm{GSN} = \left\| \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{e}} \right\|_2 \,, \tag{17.12}$$

where $\boldsymbol{e}$ is the set of disturbances and $\boldsymbol{g}$ is the gait indicator which characterizes the *failure mode*, motion finally end up a tumble. For a 2-D limit cycle walker, they took floor irregularities (step height to walk over) and step time as $\boldsymbol{e}$ and $\boldsymbol{g}$, respectively.

For real robots, GSN can be experimentally obtained from the response of gait indicators $\boldsymbol{g}$ to a single

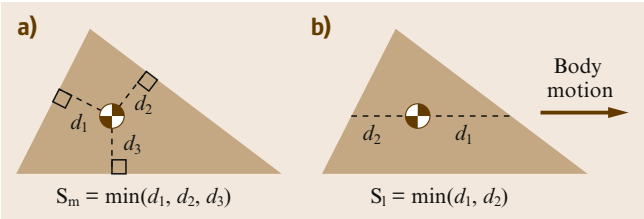


Fig.17.38a,b Definition of stability margins. **(a)** Stability margin; **(b)** Longitudinal stability margin

disturbance e_0 as

$$\left\| \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{e}} \right\|_2 = \frac{1}{|e_0|} \sqrt{\sum_{i=1}^{q} \sum_{k=0}^{\infty} [\boldsymbol{g}_k(i) - \boldsymbol{g}^*(i)]^2} \,, \qquad (17.13)$$

where $\boldsymbol{g}_k(i)$ is the i-th gait indicator value at k steps after the disturbance e_0 has applied, q is the number of gait indicators. GSN has a good correlation with the basin of attraction, hence it can be used for a design of a robust walking robot and its controller.

17.6.3 Duty Factor and Froude Number

Throughout this chapter, we have observed various walking robots which might fit best for certain environment and purpose. In some cases, however, we need to compare walking robots which have different masses, sizes, and leg numbers by using a certain performance index. Such an index should be dimensionless like a Mach number or a Raynords number in fluid mechanics.

One of the useful indices for walking machines is a duty factor β defined as

$$\beta = \frac{\text{(support period)}}{\text{(cycle time)}} \,.$$

Duty factors can be used to make the distinction between walks and runs, since we have $\beta \geq 0.5$ for walking and $\beta < 0.5$ for running [17.87].

Froude number is used in fluid mechanics to explain the behavior of surface waves. Since both of surface waves and legged locomotion are dynamic motion in gravity, *Alexander* used it to characterize animal locomotion [17.87, 88]. He calculated a Froude number by

$$Fr_2 = \frac{V^2}{gh} \,, \qquad (17.14)$$

where V is the walking or running speed, g is the gravity acceleration, and h is the height of hip joint from the ground. He showed animals of different sizes uses similar gaits when they travel with equal Froude numbers. Especially, most animals changes

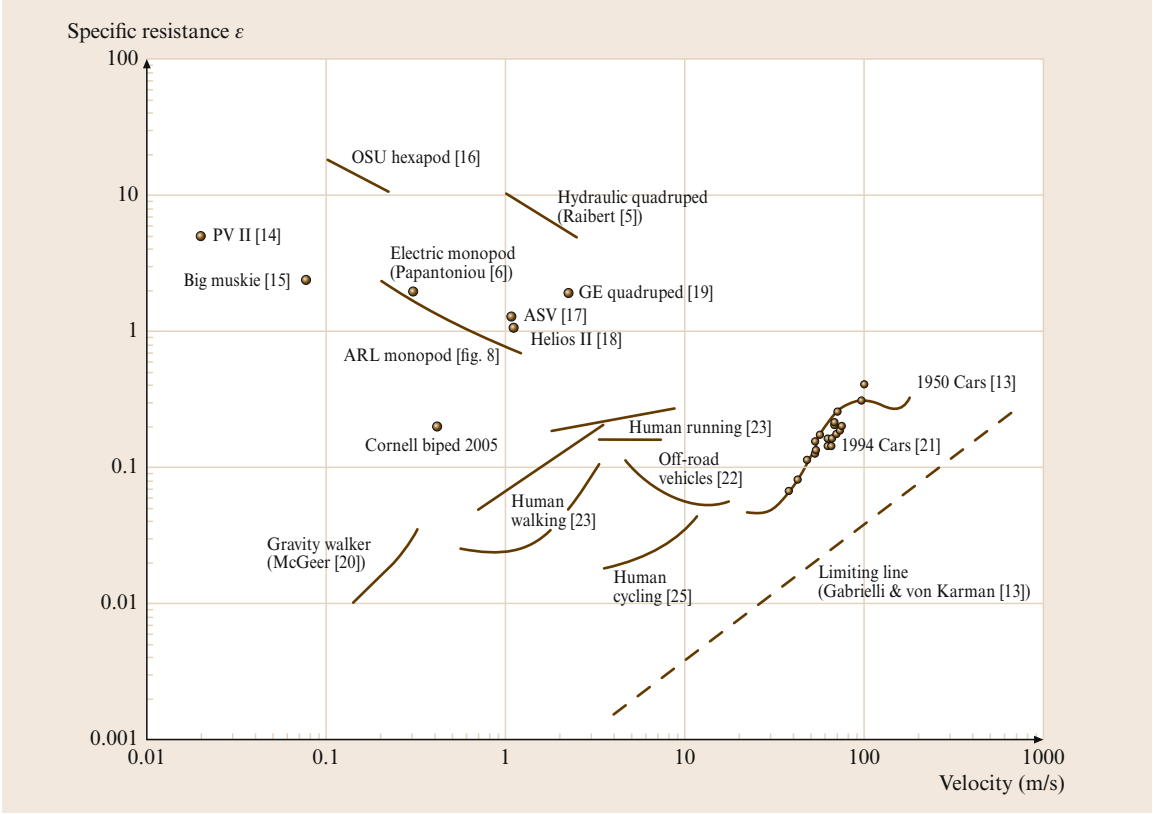


Fig. 17.39 Gabrielli–von Karman diagram (after [17.86])

their gait from walking to running at the speed of $Fr_2 = 1$.

Froude number is also defined as

$$Fr_1 = \frac{V}{\sqrt{gh}}\,, \tag{17.15}$$

which is the square root of Fr_2 and can be used non dimensional speed of animals or legged robots.

17.6.4 Specific Resistance

The *specific resistance* is another important dimensionless number, which is used to evaluate the energy efficiency of a mobile robot.

Gabrielli and von Karman discussed the performance of various vehicles using the power consumption per unit distance. That is

$$\epsilon = \frac{E}{Mgd}\,, \tag{17.16}$$

where E is the total energy consumption for a travel of distance d, M is the total mass of the vehicle and g is gravity acceleration [17.89]. Note that when we push a box of mass M for a distance d on a floor with a friction coefficient μ, we consume the power $Mg\mu d$ and the specific resistance becomes $\epsilon = \mu$. Therefore, we can say that the specific resistance indicates how smooth the locomotion is.

In the original work, Gabrielli and von Karman plotted the specific resistance as the function of speed for various vehicles (Fig. 17.39). This is called the *Gabrielli–von Karman diagram* and it was used to compare various styles of locomotion by *Umetani* and *Hirose* [17.90]. *Gregorio* et al. also showed the specific resistance of recent walking robots including their efficient hopping robot, ARL monopod [17.86]. *Collins* et al. developed robots based on passive-dynamics with small actuators and achieved energy efficiency close to human walking [17.91]. In the paper, the specific resistance is referred to as *specific energetic cost of transport*.

Video-References

VIDEO 512 Linear inverted pendulum mode
available from http://handbookofrobotics.org/view-chapter/17/videodetails/512

VIDEO 517 Hexapod robot *Ambler*
available from http://handbookofrobotics.org/view-chapter/17/videodetails/517

VIDEO 520 Hexapod ParaWalker-II
available from http://handbookofrobotics.org/view-chapter/17/videodetails/520

VIDEO 521 Cockroach-like hexapod
available from http://handbookofrobotics.org/view-chapter/17/videodetails/521

VIDEO 522 Bipedal humanoid robot: WABIAN
available from http://handbookofrobotics.org/view-chapter/17/videodetails/522

VIDEO 522 Cybernetic human HRP-4C walking
available from http://handbookofrobotics.org/view-chapter/17/videodetails/524

VIDEO 525 Cybernetic human HRP-4C quick turn
available from http://handbookofrobotics.org/view-chapter/17/videodetails/525

VIDEO 526 Development of a humanoid robot DARwIn
available from http://handbookofrobotics.org/view-chapter/17/videodetails/526

VIDEO 527 Passive dynamic walking with knees
available from http://handbookofrobotics.org/view-chapter/17/videodetails/527

VIDEO 529 Intuitive control of a planar bipedal walking robot
available from http://handbookofrobotics.org/view-chapter/17/videodetails/529

VIDEO 530 IHMC/Yobotics biped
available from http://handbookofrobotics.org/view-chapter/17/videodetails/530

VIDEO 531 Torque controlled humanoid robot TORO
available from http://handbookofrobotics.org/view-chapter/17/videodetails/531

VIDEO 532 3-D passive dynamic walking robot
available from http://handbookofrobotics.org/view-chapter/17/videodetails/532

VIDEO 533 Biped running robot MABEL
available from http://handbookofrobotics.org/view-chapter/17/videodetails/533

VIDEO 534 STriDER: Self-excited tripedal dynamic experimental robot
available from http://handbookofrobotics.org/view-chapter/17/videodetails/534

VIDEO 535 Roller-Walker: Leg-wheel hybrid vehicle
available from http://handbookofrobotics.org/view-chapter/17/videodetails/535

Avidbots Ex1016
Page 480 of 2154

VIDEO 536 RHex rough-terrain robot
available from http://handbookofrobotics.org/view-chapter/17/videodetails/536
VIDEO 537 Whegs II: A mobile robot using abstracted biological principles
available from http://handbookofrobotics.org/view-chapter/17/videodetails/537
VIDEO 540 StickybotIII climbing robot
available from http://handbookofrobotics.org/view-chapter/17/videodetails/540
VIDEO 541 Waalbot: agile climbing with synthetic fibrillar dry adhesives
available from http://handbookofrobotics.org/view-chapter/17/videodetails/541

References

17.1 M.H. Raibert: *Legged Robots That Balance* (MIT Press, Cambridge 1986)
17.2 S.-M. Song, K.J. Waldron: *Machines That Walk: The Adaptive Suspension Vehicle* (MIT Press, Cambridge 1989)
17.3 S. Lohmeier: Design and Realization of a Humanoid Robot for Fast and Autonomous Bipedal Locomotion (Technische Universität München, München 2010)
17.4 R.D. Quinn, R.E. Ritzmann: Construction of a hexapod robot with cockroach kinematics benefits both robotics and biology, Connect. Sci. **10**(3), 239–254 (1998)
17.5 H. Hirukawa, F. Kanehiro, K. Kaneko, S. Kajita, M. Morisawa: Dinosaur robotics for entertainment applications, IEEE Robotics Autom. Mag. **14**(3), 43–51 (2007)
17.6 D.E. Koditschek, R.J. Full, M. Buehler: Mechanical aspects of legged locomotion control, Arthropod Struct. Dev. **33**, 251–272 (2004)
17.7 R. Tajima, K. Suga: Motion having a flight phase: Experiments involving a one-legged robot, Proc. Int. Conf. Intell. Robots Syst. (IROS), Beijing (2006) pp. 1727–1731
17.8 K. Kaneko, S. Kajita, F. Kanehiro, K. Yokoi, K. Fujiwara, H. Hirukawa, T. Kawasaki, M. Hirata, T. Isozumi: Design of advanced leg module for humanoid robotics project of METI, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (2002) pp. 38–45
17.9 Y. Suhagara, H. Lim, T. Hosobata, Y. Mikuriya, H. Sunazuka, A. Takanishi: Realization of dynamic human-carrying walking by a biped locomotor, Proc. IEEE Int. Conf. Robotics Autom. (ICRA), New Orleans (2004) pp. 3055–3060
17.10 S. Hirose, T. Masui, H. Kikuchi, Y. Fukuda, Y. Umetani: TITAN III: A quadruped walking vehicle – Its structure and basic characteristics, Proc. Int. Symp. Robotics Res., Kyoto (1984) pp. 325–331
17.11 M. Buehler, R. Playter, M. Raibert: Robots step outside, Proc. Int. Symp. Adapt. Motion Anim. Mach. (AMAM), Ilmenau (2005)
17.12 K.J. Waldron, R.B. McGhee: The adaptive suspension vehicle, IEEE Control Syst. Mag. **6**, 7–12 (1986)
17.13 R.A. Brooks: A robot that walks; Emergent behavior from a carefully evolved network, Proc. IEEE Int. Conf. Robotics Autom. (ICRA), Scottsdale (1989) pp. 292–296
17.14 M. Hirose, Y. Haikawa, T. Takenaka, K. Hirai: Development of humanoid robot ASIMO, Proc. Int. Conf. Intell. Robots Syst. (IROS) – Workshop 2 (2001)
17.15 K. Kaneko, F. Kanehiro, M. Morisawa, K. Miura, S. Nakaoka, S. Kajita: Cybernetic Human HRP-4C, IEEE-RAS Int. Conf. Humanoid Robots, Paris (2009) pp. 7–14
17.16 K. Kaneko, F. Kanehiro, M. Morisawa, T. Tsuji, K. Mira, S. Nakaoka, S. Kajita, K. Yokoi: Hardware improvement of cybernetic human HRP-4C towards entertainent use, Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS), San Fransisco (2011) pp. 4392–4399
17.17 K. Kaneko, F. Kanehiro, S. Kajita, H. Hirukawa, T. Kawasaki, M. Hirata, K. Akachi, T. Isozumi: Humanoid Robot HRP-2, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (2004) pp. 1083–1090
17.18 K. Kaneko, K. Harada, F. Kanehiro, G. Miyamori, K. Akachi: Humanoid Robot HRP-3, Proc. Int. Conf. Intell. Robots Syst. (IROS) (2008) pp. 2471–2478
17.19 M. Kouchi, M. Mochimaru, H. Iwasawa, S. Mitani: Anthropometric database for Japanese population 1997-98, Japanese Industrial Standards Center, AIST, MITI http://riodb.ibase.aist.go.jp/dhbodydb/ (Tokyo 2000)
17.20 K. Kaneko, K. Harada, F. Kanehiro: Development of multi-fingered hand for life-size humanoid robots, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (2007) pp. 913–920
17.21 Q. Lu, C. Ortega, O. Ma: Passive gravity compensation mechanisms: Technologies and applications, Recent Pat. Eng. **5**(1), 32–44 (2011)
17.22 K.A. Wyrobek, E.H. Berger, H.F.M. Van der Loos, J.K. Salisbury: Towards a personal robotics development platform: Rationale and design of an intrinsically safe personal robot, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (2008)
17.23 Willow Garage Inc., 68 Willow Road, Menlo Park, CA 94025, USA: http://www.willowgarage.com/pages/pr2/
17.24 T. McGeer: Passive dynamic walking, Int. J. Robotics Res. **9**(2), 62–82 (1990)
17.25 M. Garcia, A. Chatterjee, A. Ruina, M. Coleman: The simplest walking model: Stability, complexity, and scaling, ASME J. Biomech. Eng. **120**, 281–288 (1998)
17.26 A. Goswami, B. Thuilot, B. Espiau: A study of the passive gait of a compass-like biped robot: Symmetry and chaos, Int. J. Robotics Res. **17**, 1282–1301 (1998)

17.27 S.H. Collins, M. Wisse, A. Ruina: A three-dimensional passive-dynamic walking robot with two legs and knees, Int. J. Robotics Res. **20**(2), 607–615 (2001)
17.28 S.H. Collins, A. Ruina, R. Tedrake, M. Wisse: Efficient bipedal robots based on passive dynamic walkers, Sci. Mag. **307**, 1082–1085 (2005)
17.29 P.A. Bhounsule, J. Cortell, A. Ruina: Design and control of Ranger: an energy-efficient, dynamic walking robot, 15th Int. Conf. Climb. Walk. Robots (CLAWAR), Baltimore (2012) pp. 441–448
17.30 S.H. Collins, M. Wisse, A. Ruina: Three-dimensional passive-dynamic walking robot with two legs and knees, Int. J. Robotics Res. **20**(2), 607–615 (2001)
17.31 M. Wisse, L. Schwab, F.L.T. Van der Helm: Passive walking dynamic model with upper body, Robotica **22**(6), 681–688 (2004)
17.32 E.R. Westervelt, J.W. Grizzle, D.E. Koditschek: Hybrid zero dynamics of planar biped walkers, IEEE Trans. Autom. Control **48**(1), 42–56 (2003)
17.33 E.R. Westervelt, J.W. Grizzle, C. Chevallereau, J.H. Choi, B. Morris: *Feedback Control of Dynamic Bipedal Robot Locomotion* (CRC, Boca Raton 2007)
17.34 C. Chevallereau, G. Abba, Y. Aoustin, F. Plestan, E.R. Westervelt, C. Canudas-de-Wit, J.W. Grizzle: RABBIT: A testbed for advanced control theory, IEEE Control Syst. Mag. **23**(5), 57–79 (2003)
17.35 J.W. Grizzle, J. Hurst, B. Morris, H.W. Park, K. Sreenath: MABEL, A new robotic bipedal walker and runner, Proc. IEEE Am. Control Conf. (2009)
17.36 D. Hobbelen, T. de Boer, M. Wisse: System overview of bipedal robots Flame and TUlip: Tailor-made for Limit Cycle Walking, Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS), Nice (2008) pp. 2486–2491
17.37 R.J. Full, D.E. Koditschek: Templates and anchors: neuromechanical hypotheses of legged locomotion on land, J. Exp. Biol. **202**, 3325–3332 (1999)
17.38 S. Kajita, K. Tani: Study of dynamic biped locomotion on rugged terrain – Derivation and application of the linear inverted pendulum mode, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (1991) pp. 1405–1411
17.39 M.B. Popovic, A. Goswami, H. Herr: Angular momentum regulation during human walking: Biomechanics and control, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (2004) pp. 2405–2411
17.40 M.B. Popovic, A. Goswami, H. Herr: Ground reference points in legged locomotion: Definitions, biological trajectories and control implications, Int. J. Robotics Res. **24**(12), 1013–1032 (2005)
17.41 S. Kajita, F. Kanehiro, K. Kaneko, K. Fujiwara, K. Harada, K. Yokoi, H. Hirukawa: Biped walking pattern generation by using preview control of zero-moment point, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (2003) pp. 1620–1626
17.42 Y. Choi, D. Kim, Y. Oh, B.J. You: Posture/walking control for humanoid robot based on kinematic resolution of com Jacobian with embedded motion, IEEE Trans. Robotics **23**(6), 1285–1293 (2007)
17.43 J. Englsberger, C. Ott, M. Roa, A. Albu-Schaeffer, G. Hirzinger: Bipedal walking control based on capture point dynamics, Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS) (2011) pp. 4420–4427
17.44 S. Lohmeier, T. Bushmann, H. Ulbrich: Humanoid Robot LOLA, Proc. IEEE Int. Conf. Robotics Autom. (ICRA), Kobe (2009) pp. 775–780
17.45 I.W. Park, J.-Y. Kim, J. Lee, J.H. Oh: Mechanical design of humanoid robot platform KHR-3 (KAIST Humanoid Robot 3: HUBO), Proc. IEEE-RAS Int. Conf. Humanoid Robots (2005) pp. 321–326
17.46 B. Stephens: Humanoid push recovery, Proc. IEEE-RAS Int. Conf. Humanoid Robots (2007)
17.47 T. Takenaka, T. Matsumoto, T. Yoshiike: Real time motion generation and control for biped robot – 1st Report: Walking gait pattern generation, Proc. IEEE /RSJ Int. Conf. Intell. Robots Syst. (IROS) (2009) pp. 1084–1091
17.48 R. Blickhan: The spring mass model for running and hopping, J. Biomech. **22**(11–12), 1217–1227 (1989)
17.49 H. Geyer, A. Seyfarth, R. Blickhan: Compliant leg behaviour explains basic dynamics of walking and running, Proc. Biol. Sci. **273**(1603), 2861–2867 (2006)
17.50 J. Pratt, G. Pratt: Exploiting natural dynamics in the control of a planar bipedal walking robot, Proc. 36th Ann. Allerton Conf. Commun. (1998)
17.51 J. Pratt, B. Krupp: Design of a bipedal walking robot, SPIE Def. Sec. Symp., Bellingham (2008)
17.52 G. Hirzinger, N. Sporer, A. Albu-Schaeffer, M. Haehnle, R. Krenn, A. Pascucci, M. Schedl: DLR's torque-controlled light weight robot III – are we reaching the technological limits now?, Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS) (2002) pp. 1710–1716
17.53 C. Ott, O. Eiberger, W. Friedl, B. Baeuml, U. Hillenbrand, C. Borst, A. Albu-Schaeffer, B. Brunner, H. Hirschmueller, S. Kielhoefer, R. Konietschke, M. Suppa, T. Wimboeck, F. Zacharias, G. Hirzinger: A humanoid two-arm system for dexterous manipulation, Proc. IEEE-RAS Int. Conf. Humanoid Robots, Genova (2006) pp. 276–283
17.54 C. Ott, C. Baumgaertner, J. Mayr, M. Fuchs, R. Burger, D. Lee, O. Eiberger, A. Albu-Schaeffer, M. Grebenstein, G. Hirzinger: Development of a biped robot with torque controlled joints, Proc. IEEE-RAS Int. Conf. Humanoid Robots (2010) pp. 167–173
17.55 G.A. Pratt, M.M. Williamson: Series elastic actuators, IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS) (1995) pp. 399–406
17.56 R. Brooks, C. Breazeal, M. Marjanovic, B. Scassellati, M. Williamson: The Cog project: Building a humanoid robot, Lect. Notes Comput. Sci. **1562**, 52–87 (1999)
17.57 H. Iwata, S. Sugano: Development of human symbiotic robot: WENDY, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (1999)
17.58 M.W. Spong: Modeling and control of elastic joint robots, Trans. ASME: J. Dyn. Syst. Meas. Control **109**, 310–318 (1987)
17.59 G. Wyeth: Control issues for velocity sourced series elastic actuators, Proc. Australasian Conf. Robotics Autom. (2006)
17.60 H. Vallery, R. Ekkelenkamp, H. van der Kooij, M. Buss: Passive and accurate torque control of se-

Avidbots Ex1016
Page 482 of 2154

ries elastic actuators, IEEE/RSJ Proc. Int. Conf. Intell. Robots Syst. (IROS) (2007)
17.61 C. Ott, A. Albu-Schaeffer, G. Hirzinger: Decoupling based cartesian impedance control of flexible joint robots, IEEE Int. Conf. Robotics Autom. (ICRA) (2003)
17.62 C. Ott, A. Albu-Schaeffer, A. Kugi, G. Hirzinger: On the passivity based impedance control of flexible joint robots, IEEE Trans. Robotics **24**(2), 416–429 (2008)
17.63 J. Heaston, D. Hong, I. Morazzani, P. Ren, G. Goldman: STriDER: Self-excited tripedal dynamic experimental robot, Proc. IEEE Int. Conf. Robotics Autom. (ICRA), Roma (2007) pp. 2776–2777
17.64 A. Rachmat, A. Besari, R. Zamri, A. Satria Prabuwono, S. Kuswadi: The study on optimal gait for five-legged robot with reinforcement learning, Int. Conf. Intell. Robots Appl. (2009) pp. 1170–1175
17.65 O. Matsumoto, S. Kajita, M. Saigo, K. Tani: Dynamic trajectory control of passing over stairs by a biped type leg-wheeled robot with nominal reference of static gait, Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS) (1998) pp. 406–412
17.66 S. Hirose, H. Takeuchi: Study on roller-walk (basic characteristics and its control), Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (1996) pp. 3265–3270
17.67 U. Saranli, M. Buehler, D.E. Koditschek: RHex: A Simple and Highly Mobile Hexapod Robot, Int. J. Robotics Res. **20**(7), 616–631 (2001)
17.68 T.J. Allen, R.D. Quinn, R.J. Bachmann, R.E. Ritzmann: Abstracted biological principles applied with reduced actuation improve mobility of legged vehicles, Proc. IEEE Int. Conf. Intell. Robots Syst. (IROS), Las Vegas (2003) pp. 1370–1375
17.69 G. Endo, S. Hirose: Study on roller-walker: System integration and basic experiments, Proc. IEEE Int. Conf. Robotics Autom. (ICRA), Detroit (1999) pp. 2032–2037
17.70 N. Neville, M. Buehler, I. Sharf: A bipedal running robot with one actuator per leg, Proc. IEEE Int. Conf. Robotics Autom. (ICRA), Orlando (2006) pp. 848–853
17.71 J. Bares, D. Wettergreen: Dante II: Technical description, results and lessons learned, Int. J. Robotics Res. **18**(7), 621–649 (1999)
17.72 N. Koyachi, H. Adachi, M. Izumi, T. Hirose, N. Senjo, R. Murata, T. Arai: Multimodal control of hexapod mobile manipulator MELMANTIS-1, Proc. 5th Int. Conf. Climb. Walk. Robots (2002) pp. 471–478
17.73 Y. Ota, T. Tamaki, K. Yoneda, S. Hirose: Development of walking manipulator with versatile locomotion, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (2003) pp. 477–483
17.74 S. Hirose, K. Yoneda, H. Tsukagoshi: TITAN VII: Quadruped walking and manipulating robot on a steep slope, IEEE Int. Conf. Robotics Autom. (ICRA), Albuquerque (1997) pp. 494–500
17.75 S. Hirose, A. Nagakubo, R. Toyama: Machine that can walk and climb on floors, walls and ceilings, Proc. 5th Int. Conf. Adv. Robotics (ICAR), Pisa (1991) pp. 753–758
17.76 T. Yano, S. Numao, Y. Kitamura: Development of a self-contained wall climbing robot with scanning type suction cups, Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS), Vol. 1 (1998) pp. 249–254
17.77 S. Kim, A. Asbeck, W. Provancher, M.R. Cutkosky: SpinybotII: Climbing hard walls with compliant microspines, Proc. Int. Conf. Adv. Robotics (ICAR), Seattle (2005) pp. 18–20
17.78 A.T. Asbeck, S. Kim, A. McClung, A. Parness, M.R. Cutkosky: Climbing walls with microspines (Video), Proc. IEEE Int. Conf. Robotics Autom. (ICRA), Orlando (2006)
17.79 R.B. McGhee, A.A. Frank: On the stability properties of quadruped creeping gaits, Math. Biosci. **3**, 331–351 (1968)
17.80 R.B. McGhee: Vehicular legged locomotion. In: *Advances in Automation and Robotics*, ed. by G.N. Saridis (JAI, Greenwich 1985) pp. 259–284
17.81 M. Vukobratović, J. Stepanenko: On the stability of anthropomorphic systems, Math. Biosci. **15**, 1–37 (1972)
17.82 Q. Huang, K. Yokoi, S. Kajita, K. Kaneko, H. Arai, N. Koyachi, K. Tanie: Planning walking patterns for a biped robot, IEEE Trans. Robotics Autom. **17**(3), 280–289 (2001)
17.83 D.A. Messuri, C.A. Klein: Automatic body regulation for maintaining stability of a legged vehicle during rough-terrain locomotion, IEEE J. Robotics Autom. **RA-1**(3), 132–141 (1985)
17.84 E. Garcia, P. de Gonzalez Santos: An improved energy stability margin for walking machines subject to dynamic effects, Robotica **23**(1), 13–20 (2005)
17.85 D.G.E. Hobbelen, M. Wisse: A disturbance rejection measure for limit cycle walkers: The gait sensitivity norm, IEEE Trans. Robotics **23**(6), 1213–1224 (2007)
17.86 P. Gregorio, M. Ahmadi, M. Buehler: Design, control, and energetics of an electrically actuated legged robot, IEEE Trans. Syst. Man Cybern. **B27**(4), 626–634 (1997)
17.87 R. McNeill Alexander: The gait of bipedal and quadrupedal animals, Int. J. Robotics Res. **3**(2), 49–59 (1984)
17.88 R. McNeill Alexander: *Exploring Biomechanics – Animals in Motion* (Freeman, Boston 1992)
17.89 G. Gabrielli, T. von Karman: What price speed – Specific power required for propulsion of vehicles, Mechan. Eng. **72**(10), 775–781 (1950)
17.90 Y. Umetani, S. Hirose: Biomechanical study on serpentine locomotion – Mechanical analysis and zoological experiment for the stationary straightforward movement, Trans. Soc. Instrum. Control Eng. **6**, 724–731 (1973), in Japanese
17.91 S. Collins, A. Ruina, R. Tedrake, M. Wisse: Efficient Bipedal Robots Based on Passive-Dynamic Walkers, Science **307**, 1082–1085 (2005)

Avidbots Ex1016
Page 483 of 2154

Multimedia Contents

18. Parallel Mechanisms

Jean-Pierre Merlet, Clément Gosselin, Tian Huang

This chapter presents an introduction to the kinematics and dynamics of parallel mechanisms, also referred to as parallel robots. As opposed to classical serial manipulators, the kinematic architecture of parallel robots includes *closed-loop kinematic chains*. As a consequence, their analysis differs considerably from that of their serial counterparts. This chapter aims at presenting the fundamental formulations and techniques used in their analysis.

18.1 **Definitions** 443
18.2 **Type Synthesis of Parallel Mechanisms** 445
18.3 **Kinematics** 446
18.3.1 Inverse Kinematics 446
18.3.2 Forward Kinematics 446
18.4 **Velocity and Accuracy Analysis** 447
18.5 **Singularity Analysis** 448
18.5.1 Parallel Singularity Analysis 448
18.6 **Workspace Analysis** 450
18.7 **Static Analysis** 451
18.8 **Dynamic Analysis** 452
18.9 **Design** 452
18.10 **Wire-Driven Parallel Robots** 453
18.10.1 Cable-Suspended Parallel Mechanisms 454
18.10.2 Fully Constrained Wire-Driven Parallel Mechanisms 454
18.10.3 Design and Prototyping 455
18.11 **Application Examples** 455
18.12 **Conclusion and Further Reading** 455
Video-References 456
References 456

18.1 Definitions

A *closed-loop kinematic chain* is one in which the links and joints are arranged such that at least one closed loop exists. Furthermore, a *complex closed-loop kinematic chain* is obtained when one of the links – other than the base – has a degree of connectivity greater than or equal to 3, i. e., one link other than the base is connected through joints to at least three other links. A *parallel manipulator* can be defined as a closed-loop mechanism composed of an end effector having n degrees of freedom (DOF) and a fixed base, linked together by at least two independent kinematic chains.

An example of such a structure was patented in 1928 by *Gwinnett* [18.1] to be used as a platform for a movie theatre. In 1947, *Gough* [18.2] established the basic principles of a mechanism with a closed-loop kinematic structure (Fig. 18.1) that allows the positioning and the orientation of a moving platform so as to test tire wear and tear. He built a prototype of this machine in 1955.

For this mechanism, the moving effector is a hexagonal platform whose vertices are all connected to a link by a ball-and-socket (spherical) joint. The other end of the link is attached to the base by a universal joint. A linear actuator allows the modification of the total length of the link; this mechanism is, therefore, a closed-loop kinematic structure, actuated by six linear actuators.

In 1965, *Stewart* [18.3] suggested the use of such a structure for flight simulators and the Gough mechanism is sometimes referred to as the *Stewart platform*

Avidbots Ex1016
Page 484 of 2154

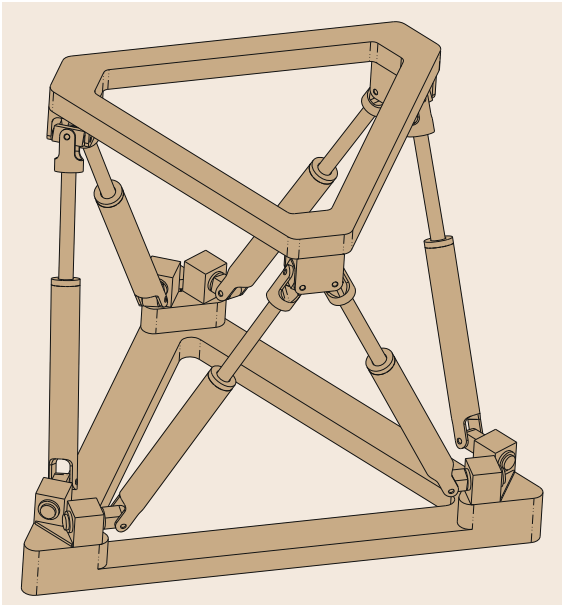


Fig. 18.1 Gough platform

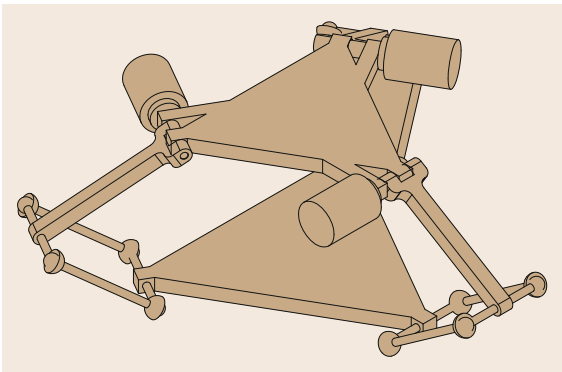


Fig. 18.2 The Delta robot

or the *Gough–Stewart platform*. The same architecture was also concurrently proposed by *Kappel* as a motion simulation mechanism [18.4]. Nowadays, the Gough platform and its variants are the platform of choice for many applications, for example, flight simulators. This application provides a convincing illustration of the main advantage of parallel robots namely, their load-carrying capacity. Indeed, while the ratio of the mass of the payload over the mass of the robot is typically smaller than 0.15 for serial 6R industrial robots, this ratio can be larger than 10 for parallel structures. Another advantage of the Gough platform is its very good positioning accuracy (although it has been questioned [18.5]). This high accuracy arises from the fact that the legs work essentially in tension/compression and are subjected to virtually no bending – thereby

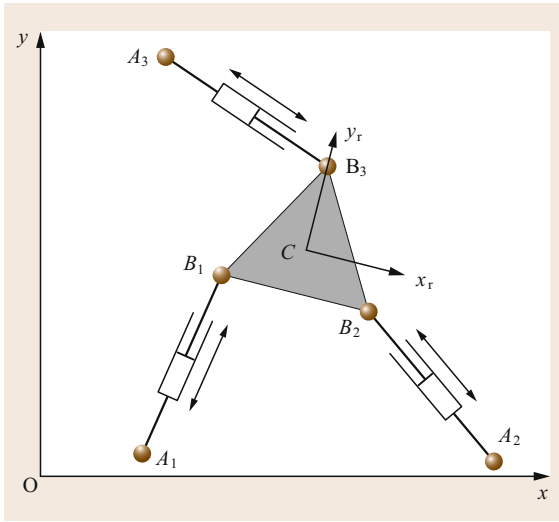


Fig. 18.3 The $3-\underline{RPR}$ planar parallel robot

leading to very small deformations – and from the fact that the errors in the internal sensors of the robot (measurement of the lengths of the legs for the Gough platform) are mapped into very small errors of the platform position. Parallel robots are also almost insensitive to scaling (the same structure can be used for large or micro robots) and they can be built using almost any type of joint, actuator, or transmission, including exotic ones such as tape spring and flexure joints [18.6] or binary actuators [18.7]. The main drawbacks of parallel robots are their small workspace and the singularities that can appear within this workspace. However, large workspaces may be obtained using wire transmissions (see, for example, the Robocrane [18.8]): such robots will be specifically addressed in Sect. 18.10 as their analysis is somewhat different. Apart from the Gough platform, the most successful designs of parallel robots are the Delta robot proposed by *Clavel* [18.9] (Fig. 18.2) and some planar parallel robots. The most common planar parallel robots have three identical legs each having an $\underline{RPR}$ or $\underline{RRR}$ architecture, where the underlined joint is the actuated one (such robots are often denoted as $3-\underline{RPR}$ ($3-\underline{RRR}$; Fig. 18.3).

The geometric arrangement of the joints and links of the Delta structure provides three translational degrees of freedom at the platform. Numerous other types of parallel robots have been proposed in the recent years. Although most existing architectures are based on the intuition of their designer, the synthesis of parallel mechanisms can be dealt with systematically. An overview of the main approaches to the type synthesis of parallel mechanisms is given in the next section.

18.2 Type Synthesis of Parallel Mechanisms

Determining all potential mechanical architectures of parallel robots that can produce a given motion pattern at the moving platform is a challenging problem. Several authors have addressed this problem, referred to as type synthesis. The approaches proposed in the literature can be divided into four groups:

- *Approaches based on graph theory*: The enumeration of all possible structures having a given number of DOFs can be performed by considering that there is only a finite set of possible kinematic pairs, and hence a very large, but finite, set of possible structure combinations (e.g., [18.10]). Classical mobility formulae – for example, the Chebychev–Grübler–Kutzbach formula – are then used to determine the number of DOFs in the structure. Unfortunately, these formulas do not take into account geometric properties of the mechanism that can change the number of DOFs of the platform. Although many parallel robots have been successfully invented and developed for industrial uses, based on graph theory – the Neos Tricept [18.11] and the Sprint Z3 head of DS Technologie Gmbh, for example – the synthesis approaches based strictly on graph theory can produce only limited nonoverconstrained parallel robots and have been largely superseded by the other methods described below.
- *Approaches based on group theory*: The motion of a rigid body has the special structure of a group, the *displacement group*. Subgroups of the group of displacements, such as the spatial translations or all the translational and rotational motions about all axes that are parallel to the axis defined by a given vector (Schönflies motion [18.12]), play an important role as they can be combined through the *intersection* operation [18.13] when elements of subgroups act on the same rigid body. Type synthesis consists in determining all the possible subgroups to which the different kinematic chains that will constitute the legs of the robot may belong to so that their intersection leads to the desired motion pattern for the platform. The synthesis approaches based on group theory led to the discovery of numerous possible architectures [18.14]. Nevertheless, the displacement group has special properties that are not reflected by its group structure alone. Also, the approach is limited to motion patterns that can be described by a subgroup of the displacement group.
- *Approach based on differential geometry*: As a full extension and completion of the Lie-group-algebraic method, the differential geometry method can be used to cover the motion patterns that cannot be modeled by subgroups but regular submanifolds, a plane-hinged parallelogram, for example [18.15]. The additional merit of the geometric method lies in that it enables to link the finite motion with the instantaneous motion of the platform thanks to the integral/differential relationship between the Lie group and Lie algebra.
- *Approaches based on screw theory*: In this approach, the first step is to determine the wrench system S that is reciprocal to the theoretically accessible instantaneous twist (or velocity twist) of the moving platform. Then, the wrenches of each limb whose union spans the system S (and that determine all the possible structures of the kinematic chains that will generate the corresponding wrenches) are enumerated. Although this method is very effective at providing simple and intuitive descriptions of the instantaneous motions of the platform, it is necessary to verify that the mobility of the platform is full cycle and not only instantaneous as all considered twists and wrenches are instantaneous. Thus, techniques to ensure the consistency between the finite and instantaneous motions have also been developed. A systematic implementation of this approach is provided in [18.16].

The above synthesis methods have been used to generate a large number of architectures that cannot be presented in this book, but the website [18.17] presents a comprehensive description of a large number of mechanical architectures. Among others, several remarkable architectures were proposed in the recent years to generate Schönflies motions using the innovative concept of articulated plates, leading to a brand new pick-and-place parallel robot known as Adept Quattro [18.18]. Also, other similar architectures allowing unlimited rotations of the end effector while avoiding all singularities have recently been proposed (see for instance [18.19]).

It should also be pointed out that although these synthesis methods can be used to generate a large number of architectures [18.20], there is a lack of a metric to evaluate which one is better than the other even if they have completely identical motion patterns. Furthermore, it is hard to link the models for type synthesis with those for kinematic, static, and dynamic analyses and design. Open issues, therefore, include:

- To develop suitable performance metrics for each motion type to identify suitable topological structures for specific applications.

Avidbots Ex1016
Page 486 of 2154

- To develop a complete theoretical package that enables type synthesis and parametric design to be integrated into a unified framework that takes into account the unavoidable uncertainties in the manufacturing process, an ultimate objective of design.

18.3 Kinematics

18.3.1 Inverse Kinematics

The solution of the inverse kinematic problem is usually straightforward for parallel robots. This can be illustrated with the Gough platform.

The solution of the inverse kinematic problem consists in determining the vector of leg lengths $\boldsymbol{q}$ for a given pose of the platform defined by the position vector $\boldsymbol{p}$ of a given point on the platform in a fixed reference frame and a rotation matrix $\mathbf{R}$ representing the orientation of the platform with respect to the fixed frame. Let $\boldsymbol{a}_i$ denote the position vector of the anchor point of the i-th leg on the base given in the fixed reference frame and $\boldsymbol{b}_i$ denote the position vector of the anchor point of the i-th leg on the platform, given in a moving frame attached to the platform. The length of the i-th leg is the norm of the vector connecting the two anchor points. This vector, noted $\boldsymbol{s}_i$, can be written as

$$\boldsymbol{s}_i = \boldsymbol{p} + \mathbf{R}\boldsymbol{b}_i - \boldsymbol{a}_i\,, \quad i = 1, \ldots, 6\,. \tag{18.1}$$

Given the pose of the platform (vector $\boldsymbol{p}$ and matrix $\mathbf{R}$), vector $\boldsymbol{s}_i$ is readily calculated using (18.1) and, hence, the leg lengths can be obtained.

18.3.2 Forward Kinematics

The solution of the forward kinematic problem consists in determining the pose of the platform for a given set of actuated joint coordinates (a given vector $\boldsymbol{q}$). This solution is needed for control purposes, calibration, and motion planning.

The forward kinematic problem of a parallel robot is usually much more complex than the inverse kinematic problem. Indeed, the loop closure equations (18.1) are typically highly nonlinear expressions of the pose variables. They form a nonlinear set of equations that generally admits multiple solutions (e.g., the Gough platform can have up to 40 solutions [18.21–23], while a table with the number of solutions for Gough platforms with special geometries is provided in [18.24]). The forward kinematic problem arises in two different contexts. In the first case, no estimate of the current pose of the platform is available (e.g., when starting the robot), while in the second case, a relatively precise estimate of the pose is known (e.g., in real-time control when the forward kinematics has been solved at the previous sampling time of the controller). In the first case, the only known approach is to determine all the solutions of the inverse kinematic equations, although there is no known algorithm to sort the set of solutions. It is often possible to determine an upper bound on the number of real solutions. Consider, for example, the case of the planar $3-\underline{R}PR$ robot (Fig. 18.3). If the joint at point B_3 is disassembled, two separate mechanisms are obtained, namely, a 4-bar linkage and a rotary lever. From the kinematics of 4-bar linkages, it is known that point B_3 moves on a sixth-order algebraic curve. Moreover, it is also known that point B_3 of the lever moves on a circle, that is, a second-order algebraic curve. For a given set of actuated joint coordinates, solutions to the forward kinematic problem arise when the two curves intersect, that is, when the mechanism can be assembled. Bezout's theorem states that algebraic curves of order m, n intersect in nm points, counting their order of multiplicity. In the above case, this means that the curves intersect in 12 points. However, these 12 points include the two circular imaginary points that belong to the coupler curve of the 4-bar linkage and to any circle, thereby to their intersection. These points are counted three times in Bezout's theorem and, hence, the forward kinematic problem will have at most six real solutions, corresponding to the intersection points.

Various methods have been proposed for the solution of the forward kinematic problem: elimination [18.25], continuation [18.23], Gröbner bases [18.26], and interval analysis [18.27]. Elimination is usually not very stable numerically (i. e., it can produce spurious roots and miss solutions) unless special care is taken in the implementation of the resulting univariate equation and the elimination steps. For instance, the transformation of polynomial solutions into eigenvalue problems can be used [18.28]. Polynomial continuation, on the other hand, is much more stable numerically since mature algorithms can be found in the literature [18.29]. The fastest methods – although not appropriate for a real-time use – are Gröbner bases and interval analysis. They also have the advantage of being numerically certified (no roots can be missed and the solution can be computed with an arbitrarily prescribed accuracy).

However, in the simplest cases, elimination can usually be used to produce stable implementations as, for example, for the $3-\underline{RPR}$ robot [18.30, 31]. Finally, it is pointed out that the forward kinematic problem solved for one type of robot may often be used for another architecture: for example, the one developed above for the $3-\underline{RPR}$ robot may be used for the $3-\underline{RRR}$. This is typical for parallel robots.

When a priori information (an initial guess of the solution) is available, the forward kinematics is usually solved using the Newton–Raphson or the Newton–Gauss iterative scheme. Consider the solution to the inverse kinematic problem, written as

$$\boldsymbol{q} = \boldsymbol{f}(\boldsymbol{x}) \,. \tag{18.2}$$

Iteration k of the Newton–Raphson procedure is written as

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \mathbf{A}(\boldsymbol{q} - \boldsymbol{f}(\boldsymbol{x}_k)) \,, \tag{18.3}$$

where $\boldsymbol{q}$ is the vector of prescribed joint variables. Matrix $\mathbf{A}$ is usually chosen as $(\partial \boldsymbol{f}/\partial q)^{-1}(\boldsymbol{x}_k)$ (the inverse matrix may not be computed at each iteration or even be chosen as constant). The iterative scheme stops when the magnitude of the difference vector $(\boldsymbol{q} - \boldsymbol{f}(\boldsymbol{x}_k))$ is smaller than a chosen threshold.

Provided that a good initial estimate of the solution is available, the Newton–Raphson algorithm is usually very fast. However, the procedure may not converge or, even worse, it may converge to a solution that is not the correct pose of the platform, that is, it may converge to another assembly mode. Such a situation can occur even if the initial guess is arbitrarily close to the correct pose. If the result is used in a control loop, the consequences can be catastrophic. Fortunately, mathematical tools, such as the Kantorovitch theorem combined with interval analysis can be used to determine if the solution found by the Newton–Raphson scheme is the correct pose of the robot, that is, to certify the result at the cost of a longer computation time, yet still compatible with real time [18.27].

Furthermore, the choice of the inverse kinematics equations plays an important role in the convergence of this method [18.27]. For example, for a Gough platform, a minimal set of equations (six equations involving six unknowns: three for the translation and three orientation angles) can be used but other representations are possible. For instance, the position coordinates of three of the platform anchor points – in the fixed reference frame – can be used as unknowns (the coordinates of the remaining three points are then readily computed upon convergence). Using this representation, nine equations are needed. Six equations are obtained based on the known distances between the anchor points on the base and platform and three additional equations are obtained based on the known distances between all combinations of the three selected platform anchor points.

Another possible approach for the solution of the forward kinematic problem is either to add sensors on the passive joints (e.g., on the U joints of a Gough platform) or to add passive legs with sensed joints. The main issue is then to determine the number and location of the additional sensors that lead to a unique solution [18.32–34] and to determine the effect of the sensor errors on the positioning error of the platform. For example, Stoughton mentions that for a Gough platform with sensors in the U joints, it is still necessary to use the Newton–Raphson scheme to improve the accuracy of the solution because the solutions obtained with the additional sensors are very sensitive to measurement noise [18.35].

18.4 Velocity and Accuracy Analysis

Similarly to serial robots, the actuated joint velocity vector of parallel robots $\dot{\boldsymbol{q}}$ is related linearly to the vector of translational and angular velocities of the platform (for simplicity, the latter vector is denoted here as $\dot{\boldsymbol{p}}$ although the angular velocity is not the time derivative of any set of angular parameters). The linear mapping between the two vectors is provided by the Jacobian matrix $\mathbf{J}$

$$\dot{\boldsymbol{p}} = \mathbf{J}(p)\dot{\boldsymbol{q}} \,. \tag{18.4}$$

However, for parallel robots, a closed-form expression is usually available for the inverse Jacobian $\mathbf{J}^{-1}$ but this is much more difficult for $\mathbf{J}$ (more precisely, the closed-form for most 6-DOF robots will be so complex that it cannot be used in practice). For example, a simple static analysis can be used to show that the i-th row $\mathbf{J}_i^{-1}$ of matrix $\mathbf{J}^{-1}$ for a Gough platform can be written as

$$\mathbf{J}_i^{-1} = (\boldsymbol{n}_i \quad \boldsymbol{c}_i \times \boldsymbol{n}_i) \,, \tag{18.5}$$

where $\boldsymbol{n}_i$ represents the unit vector defined along leg i and $\boldsymbol{c}_i$ is the vector connecting the origin of the mobile frame attached to the platform to the i-th anchor point on the platform.

Avidbots Ex1016
Page 488 of 2154

The effect of the joint sensor errors $\Delta \boldsymbol{q}$ on the positioning error $\Delta \boldsymbol{p}$ follows the same relationship, namely

$$\Delta \boldsymbol{p} = \mathbf{J}(\boldsymbol{p}) \Delta \boldsymbol{q} \,. \tag{18.6}$$

Since matrix $\mathbf{J}$ is difficult to be obtained in a closed form, accuracy analysis (i. e., finding the maximal positioning errors over a given workspace for bounded joint sensor errors) is much more difficult than for serial robots [18.36, 37]. Apart from the measurement errors, there are other sources of inaccuracy in parallel robots, namely, clearance in the passive joints, manufacturing tolerances, thermal errors, gravity induced and dynamic errors [18.38, 39]. The effect of joint clearances on trajectories followed by serial and parallel robots was studied in [18.40–42]. According to these works, it is impossible to determine trends for the effect of the geometric errors: a case-by-case study must be performed since the effect is highly dependent on the mechanical architecture, dimensioning, and workspace of the robot. Thermal effects are sometimes mentioned as possible sources of inaccuracy, although few works substantiate this claim [18.43] and cooling may slightly reduce their effects [18.44, 45].

Calibration is another means of improving the accuracy of parallel robots. This issue was addressed in Chap. 3. The methods and procedures used for parallel robots differ slightly from the ones used for serial robots since only the inverse kinematic equations are available and the positioning of the platform is much less sensitive to the geometric errors than in serial robots [18.46, 47]. Hence, the measurement noise occurring during calibration has a significant impact and may lead to surprising results. For example, the classical least-squares method may lead to parameters that are such that some constraint equations are not satisfied even when the measurement noise is taken into account [18.48] or even to parameter values that are unrealistic. It has also been shown with experimental data that classical parallel robot modeling leads to constraint equations that do not have any solution irrespective of the measurement noise [18.49]. Moreover, calibration is very sensitive to the choice of the calibration poses [18.50]: it seems that the optimal choice is to select poses on the workspace boundary [18.51, 52]. We may also mention an original calibration approach based not on the proprioceptive sensors of the robot but on leg observation with a vision system [18.53].

18.5 Singularity Analysis

The analysis of singularities of parallel mechanisms was first addressed in [18.54]. In this formulation, the kinematic equations were reduced to the input–output relationship between the actuated joint coordinate vector $\boldsymbol{q}$ and the platform Cartesian coordinate vector $\boldsymbol{p}$, namely,

$$\boldsymbol{f}(\boldsymbol{q}, \boldsymbol{p}) = 0 \,. \tag{18.7}$$

Differentiating (18.7) with respect to time leads to

$$\mathbf{B}\dot{\boldsymbol{q}} + \mathbf{A}\dot{\boldsymbol{p}} = \mathbf{0} \,. \tag{18.8}$$

Three types of singularities can then be defined: *i*) when matrix $\mathbf{B}$ is singular (termed serial singularity), *ii*) when matrix $\mathbf{A}$ is singular (termed parallel singularity) and *iii*) when the input–output equations degenerate (termed architecture singularity) in which case both matrices $\mathbf{A}$ and $\mathbf{B}$ can be singular. In a serial singular configuration, the joints can have a nonzero velocity, while the platform is at rest. In a parallel singularity, there exist nonzero platform velocities for which the joint velocities are zero. In the neighborhood of such a configuration, the robot can undergo an infinitesimal motion, while the actuators are locked. Since the mobility of the end effector should be zero when the actuators are locked, it is also said that in such a configuration, the robot *gains* some DOFs. As a consequence, certain degrees of freedom of the platform cannot be controlled, and this is a major problem.

A more general study of singularities was proposed by *Zlatanov et al.* [18.55]. The latter analysis uses the velocity equations involving the full twist of the end effector and all joint velocities (passive or actuated). This approach led to a more detailed classification of singularities and was also used later to reveal special singularities (referred to as *constraint singularities*) [18.56] that could not be found with the analysis presented in [18.54]. Also, the singularity analysis of parallel mechanisms was revisited more recently in [18.57], where hierarchical levels in which different critical phenomena originate are recognized.

The singularity analyses discussed above are of the first order. Second-(and higher-)order singularity analyses can also be performed, although these analyses are much more complex [18.42, 58].

18.5.1 Parallel Singularity Analysis

This type of singularity is especially important for parallel robots because it corresponds to configurations in

which the control of the robot is lost. Furthermore, very large joint forces can occur in the vicinity of singular poses that may lead to a breakdown of the robot. The main issues to be addressed in this context are:

1. The characterization of the singularities.
2. The definition of performance indices representing the closeness to a singularity.
3. The development of algorithms that are capable of detecting the presence of singularities over a given workspace or trajectory.

Parallel singularities arise when the 6×6 *full inverse Jacobian* $\mathbf{J}_f$ (i. e., the matrix that maps the full twist of the platform into active – and eventually passive – joint velocities) is singular, that is, when its determinant $\det(\mathbf{J}_f)$ is equal to 0. It is pointed out that passive joint velocities sometimes have to be included because restricting the analysis to the active joint velocities may not allow the determination of all singular configurations of the mechanism (see the example of the $3-U\underline{P}U$ robot [18.59]). Usually, a proper velocity analysis allows one to establish this matrix in a closed form. But computing the determinant of this matrix may be difficult even with symbolic computation tools (see the example of the Gough platform [18.60, 61]) and it does not provide an insight into the geometric conditions associated with the singularities.

An alternative approach is to use line geometry: indeed, for several parallel robots (although not all), a row of $\mathbf{J}_f$ corresponds to the Plücker vector of some line defined on the links of the robot. For example, for a Gough platform, the rows of $\mathbf{J}_f$ are the normalized Plücker vectors of the lines associated with the legs of the robot. A singularity of matrix $\mathbf{J}_f$, therefore, implies a linear dependency between these vectors (they then constitute a *linear complex*), a situation that may occur only if the lines associated with the vectors verify particular geometric constraints [18.62]. These geometric constraints have been identified by Grassmann for every set of 3, 4, 5, or 6 vectors. Singularity analysis is, thus, reduced to determining conditions on the pose parameters for which these constraints are satisfied, giving geometric information on the singularity variety. A variant of this approach is to use the Grassmann Cayley algebra that in certain cases may lead directly to the geometric singularity conditions [18.63].

Measuring closeness between a pose and a singular configuration is a difficult problem: there exists no mathematical metric defining the distance between a prescribed pose and a given singular pose. Hence, a certain level of arbitrariness must be accepted in the definition of the *distance* to a singularity and none of the proposed indices is perfect. For example, a possible index is the determinant of $\mathbf{J}_f$: unfortunately, when the platform is subjected to both translational and rotational motion, the Jacobian matrix is not dimensionally homogeneous and, hence, the value of the determinant changes according to the physical units used to describe the geometry of the robot. Dexterity indices, as defined in Chap. 10 (although they are much less relevant for parallel robots than for serial robots [18.37]), can also be used, and special indices for parallel robots have also been proposed [18.64–66]. For example, it may be interesting for a Gough platform to define a workspace as singularity-free if for any pose the absolute value of the joint forces is lower than a given threshold [18.67] or to use a physically meaningful index like the kinematic sensitivity [18.68].

Most of the analyses based on the above indices are local, that is, valid only for a given pose, while in practice the problem is to determine if a singular configuration can occur over a given workspace or trajectory. Fortunately, an algorithm exists that allows this verification, even if the geometric model of the robot is uncertain [18.69]. However, it should be pointed out that a singularity-free workspace may not always be optimal for parallel robots. Indeed, other performance requirements may impose the presence of singularities in the workspace or the robot may have part of its workspace singularity-free (for example its working region) while exhibiting singularities outside this region. Therefore, a motion planner producing a trajectory that avoids singularities while remaining close to a desired path is advisable and various approaches have been proposed to address this problem [18.70, 71]. A related problem is to determine if two solutions of the forward kinematics may be joined without crossing a singularity: this has been proved for planar robots [18.72, 73] but also for 6-DOF robots [18.74].

Finally, it is noted that being close to a singular configuration may be useful in some cases. For example, large amplification factors between the end-effector motion and the actuated joint motion may be interesting for fine positioning devices with a very small workspace or for improving the sensitivity along some measurement directions for a parallel robot used as a force sensor [18.75]. It should also be mentioned that parallel robots that remain *permanently* in a singular configuration may be interesting since they are capable of producing complex motions with only one actuator [18.76–78].

Avidbots Ex1016
Page 490 of 2154

18.6 Workspace Analysis

As mentioned earlier, one of the main drawbacks of parallel robots is their small workspace. Moreover, their workspace analysis is usually more complex than for serial robots, especially if the platform has more than 3-DOFs. Indeed, the platform motion capabilities are usually coupled, thereby prohibiting simple graphical representations of the workspace. *Decoupled robots* have also been proposed [18.79–81] (VIDEO 54 and VIDEO 52), but they do not provide the load-carrying capacity of conventional parallel robots. In general, the workspace of parallel robots is restricted by the following:

- The limitations on the actuated joint variables: for example, the lengths of the legs of a Gough platform have a minimum and maximum value.
- The limitations on the range of motion of the passive joints: for example, the U and S joints of a Gough platform have restricted ranges of angular motion.
- The mechanical interferences between the bodies of the mechanism (legs, base, and platform).

Different types of workspaces can be defined such as, for example, the *dextrous* workspace (the set of platform locations in which all orientations are possible), the *maximal* workspace (the set of platform locations that may be reached with at least one orientation), or the *orientation* workspace (the set of orientations that can be reached for a given location of the platform).

A possible approach for representing the workspace of parallel robots with $n > 3$ DOFs is to fix the value of $n-3$ pose parameters in order to be able to plot the workspace over the remaining 3-DOFs. Such plots can be obtained very efficiently using geometric methods if the plotted variables involve only translational motions because a geometric approach usually allows one to establish directly the nature of the boundary of the workspace (e.g., [18.82, 83] for the calculation of the workspace of a Gough platform when its orientation is fixed). Another advantage of this approach is that it allows the computation of the surface and volume of the workspace. Possible alternatives include:

- Discretization methods in which all poses of an n-dimensional grid are checked with respect to all types of kinematic constraints. Such methods are usually simple to implement but are computationally intensive because the computation time increases exponentially with the resolution of the mesh. Also, they require a large storage space.
- Numerical methods that allow the determination of the workspace boundary [18.84, 85].
- Numerical methods based on interval analysis that allow the determination of an approximation of the workspace volume up to an arbitrary accuracy [18.86, 87]. This representation is also appropriate for motion planning problems.

Singularities can also split the workspace calculated from the above kinematic constraints into elementary components, called the *aspects* by *Wenger* and *Chablat* [18.88] and which are separated by a singularity variety. Some very good results have been obtained for planar robots [18.89] but determining the aspects for spatial robots is still an open problem. A parallel robot may not always be able to move from one aspect to another (at least without considering the dynamics of the robot [18.90, 91]) and hence the useful workspace can be reduced.

A problem related to workspace analysis is the *motion planning problem*, which is slightly different from that encountered with serial robots. For parallel robots, the problem is not to avoid obstacles within the workspace but either to determine if a given trajectory lies entirely inside the workspace of the robot and is singularity-free (for which an algorithm is available [18.92]) or to determine such a trajectory between two poses which is much more difficult [18.93]. Indeed, classical serial robot motion planners work in the joint space and assume that there is a one-to-one relationship between the joint space and the operational space. With this assumption, it is possible to determine a set of points in the joint space that is collision-free and to use this knowledge to build a collision-free path between two poses. However, this assumption is not valid for parallel robots because the mapping between the joint space and the operational space is not one to one: a point in the joint space may either correspond to multiple points in the operational space or have no correspondence because the closure equations of the mechanism are not satisfied. For parallel robots, the most efficient motion planner seems to be an adaptation of a probabilistic motion planner that takes into account – to a certain extent – the closure equations [18.94, 95].

Another motion planning problem is related to tasks where a limited number of DOFs of the robot are used, while the others may be used to increase the machine's workspace, to avoid singularities or to optimize some performance index of the robot [18.96, 97]. Along the same line, it is possible to define the *part positioning* problem [18.98] as determining the pose of a rigid body with respect to the workspace of the robot so that this pose satisfies some constraints.

Avidbots Ex1016
Page 491 of 2154

18.7 Static Analysis

Similarly to serial robots, the static analysis of parallel robots can be readily performed using the Jacobian matrix. Indeed, the mapping between the vector of actuated joint forces/torques $\boldsymbol{\tau}$ and the external wrench $\boldsymbol{f}$ exerted by the platform can be written as

$$\boldsymbol{\tau} = \mathbf{J}^{\mathbf{T}} \boldsymbol{f} \,, \tag{18.9}$$

where $\mathbf{J}^{\mathbf{T}}$ is the transpose of the Jacobian matrix of the robot. Equation (18.9) can be used for various purposes, namely:

- During the *design* process in order to determine the maximal actuator forces/torques over the robot workspace (for actuator selection). This is a complex issue since $\mathbf{J}^{\mathbf{T}}$ is not known in a closed form.
- In applications where the robot is used as a *force sensor*: If $\boldsymbol{\tau}$ is measured and the pose of the platform is known, then (18.9) can be used to calculate $\boldsymbol{f}$ so that the robot can be used both as a motion and sensing platform [18.99–101].

The stiffness matrix $\mathbf{K}$ of a parallel robot is defined similarly to that of serial robots as

$$\mathbf{K} = \mathbf{J}^{-\mathbf{T}} \mathbf{K}_{\boldsymbol{j}} \mathbf{J}^{-\mathbf{1}} \,, \tag{18.10}$$

where $\mathbf{K}_{\boldsymbol{j}}$ is the diagonal matrix of actuated joint stiffness. *Duffy* [18.102] notes that this derivation is, in general, incomplete. For example, for a Gough platform this derivation assumes that there is no initial load on the elastic element of the link. Assuming that the length of the unloaded link is q_i^0 leads to

$$\Delta \boldsymbol{f} = \sum_{i=1}^{i=6} k \Delta q_i \boldsymbol{n_i} + k_i (q_i - q_i^0) \Delta \boldsymbol{n_i} \,,$$

$$\Delta \boldsymbol{m} = \sum_{i=1}^{i=6} k \Delta q_i \boldsymbol{c_i} \times \boldsymbol{n_i} + k_i (q_i - q_i^0) \Delta (\boldsymbol{c_i} \times \boldsymbol{n_i}) \,,$$

where k_i denotes the axial stiffness of the leg, $\boldsymbol{n_i}$ represents the unit vector defined along the i-th leg, $\boldsymbol{c_i}$ is the vector connecting the origin of the moving reference frame to the i-th anchor point of the platform, and $\boldsymbol{f}, \boldsymbol{m}$ are the external force and moment vectors that are applied by the platform. Consequently, the stiffness matrix as defined in (18.10) is valid only if $q_i = q_i^0$, and is coined the *passive stiffness*.

Moreover, it was pointed out in the literature [18.103] that the formulation of (18.10) is valid only when the external wrench is zero. Indeed, the above formulation does not account for the fact that the Jacobian matrix is configuration dependent and therefore changes with the applied loads. The formulation proposed in [18.103] and termed *conservative congruence transformation* (CCT) takes these changes into account and leads to a stiffness matrix defined as

$$\mathbf{K}_c = \mathbf{J}^{-\mathbf{T}} \mathbf{K}_{\boldsymbol{j}} \mathbf{J}^{-\mathbf{1}} + \left(\frac{\partial \mathbf{J}^{-\mathrm{T}} \boldsymbol{\tau}}{\partial \boldsymbol{p}} \right) \,, \tag{18.11}$$

where $\boldsymbol{p}$ is the vector of Cartesian coordinates and $\boldsymbol{\tau}$ is the vector of actuated joint forces/torques. Whenever possible, the formulation of (18.11) should be used instead of that of (18.10) because it is mechanically consistent.

Another interesting static problem is the static balancing of parallel robots. The static balancing of mechanisms, in general, has been an important research topic for several decades (e.g., [18.104] for an account of the state of the art and many recent new results). Parallel mechanisms are said to be *statically balanced* when the weight of the links does not produce any torque (or force) at the actuators under static conditions, for any configuration of the manipulator or mechanism. This condition is also referred to as *gravity compensation*. The gravity compensation of parallel mechanisms was addressed in [18.105] where the use of counterweights was suggested to balance a 2-DOF parallel robot used for antenna aiming, and in [18.106] for planar robots. This analysis leads to simple and elegant balancing conditions.

In general, static balancing can be achieved using counterweights and/or springs VIDEO 48. When springs are used, static balancing can be defined as the set of conditions for which the total potential energy in the mechanism – including gravitational energy and the elastic energy stored in the springs – is constant for any configuration of the mechanism. When no springs – or other means of storing elastic energy – are used, then static balancing conditions imply that the center of mass of the mechanism does not move in the direction of the gravity vector, for any motion of the mechanism.

Sufficient conditions under which a given mechanism is statically balanced may be obtained. The problem of statically balancing spatial parallel robots is usually complex [18.107, 108]). Among other results, it was pointed out that it is not possible to balance a 6-*UPS* robot in all poses by using only counterweights [18.109] (*U* stands for a Hooke joint, *P* stands for a prismatic joint and *S* stands for a spherical joint). Alternative mechanisms (using parallelograms) that can

Avidbots Ex1016
Page 492 of 2154

be balanced using only springs were suggested [18.107, 110, 111].

A natural extension of static balancing is dynamic balancing. The dynamic balancing problem consists in finding parallel mechanisms that do not produce any reaction force or torque on their base for arbitrary trajectories. This problem was addressed for planar and spatial parallel mechanisms. In [18.112], it was shown that dynamically balanced parallel mechanisms with up to 6-DOFs can be synthesized, although the resulting mechanical architectures may be complex (VIDEO 49).

18.8 Dynamic Analysis

The dynamic model of parallel robots has a form similar to that of serial robots (Chap. 3), namely,

$$\mathbf{M}(\boldsymbol{x})\ddot{\boldsymbol{p}} + \mathbf{C}(\dot{\boldsymbol{p}}, \dot{\boldsymbol{q}}, \boldsymbol{p}, \boldsymbol{q}) + \mathbf{G}(\boldsymbol{p}, \boldsymbol{q}) = \boldsymbol{\tau} \ , \qquad (18.12)$$

where $\mathbf{M}$ is the positive-definite generalized inertia matrix, $\mathbf{G}$ is the gravitational term, and $\mathbf{C}$ is the centrifugal and Coriolis term. However, this equation is more difficult to obtain for parallel robots because the closure equations have to be satisfied. A classical method for calculating the dynamic models of closed-chain mechanisms is to consider first an equivalent tree structure, and then to enforce the kinematic constraints using Lagrange multipliers or d'Alembert's principle [18.113, 114]. Other approaches include the use of the principle of virtual work [18.115–118], the Lagrange formalism [18.119–122], Hamilton's principle [18.123], and Newton–Euler equations [18.124–129].

The main drawback of dynamic models obtained for parallel robots is that they are usually rather complex, they require the determination of dynamic parameters that are often not well known (dynamic identification of closed-loop mechanisms being not trivial [18.130]), and they usually involve solving the forward kinematics. Therefore, their calculation is computer intensive while they must be used in real time.

Using dynamic models for control purposes was proposed [18.131, 132], usually in the context of an adaptive control scheme, in which the tracking errors are used on-line to correct the parameters used in the dynamic equations [18.133, 134]. Control laws were proposed mainly for planar robots and for the Delta robot [18.135, 136], although some implementations have been proposed for general 6-DOF robots [18.137, 138] or vibration platforms [18.139]. However for 6-DOF robots, the benefits of using dynamic models for high-speed motions are difficult to establish because the computational burden of the dynamic model somewhat reduces the potential gains. An original approach is to work directly in the joint space, using information obtained on the legs' orientation through a vision system, allowing to get rid of the forward kinematics [18.140].

Dynamic control is an important and open issue since parallel robots can operate at velocities and accelerations much larger than those of serial robots. For example, some of the Delta robots have reached accelerations of the order of $500\,\mathrm{m/s^2}$ (with an objective of reaching 100 G [18.141]) while wire-driven robots can probably exceed this value.

Finally, the optimization of the dynamic properties of parallel mechanisms can be addressed by minimizing the variation of the inertia over the workspace [18.142].

18.9 Design

The design of a robot can be decomposed into two main phases:

- *Structural synthesis*: Finding the general mechanical structure of the robot.
- *Geometric synthesis*: Determining the value of the geometric parameters involved in a given structure (here geometric parameters has a loose sense: for example, mass, inertia may also be involved).

The problem of structural synthesis (type synthesis) was addressed in Sect. 18.2. However, performance requirements other than the motion pattern of the robot have to be taken into account in the design of a robot. Serial robots have the advantage that the number of possible mechanical architectures is relatively small and that some of these architectures have clear advantages in terms of performance compared to others (for example the workspace of a $3R$ structure is much larger than the workspace of a Cartesian robot of similar size).

Unfortunately, no such rules exist for parallel robots, for which there are furthermore a large number of possible mechanical designs. Additionally, the performances of parallel robots are very sensitive to their geometric parameters. For example, the extremal stiffness of a Gough platform over a given workspace can change by 700% for a change of only 10% of the platform radius. Consequently, structural synthesis cannot

Avidbots Ex1016
Page 493 of 2154

be dissociated from the geometric synthesis. In fact, it is conjectured that a well-dimensioned robot of any structural type will, in general, perform better than a poorly designed robot with a structure that may seem more appropriate for the task at hand.

Usually, the design process is treated as an optimization problem. To each specified performance requirement is associated a performance index whose value increases with the level of violation of the performance requirement. These performance indices are summed in a weighted real-value function, called the *cost function*, which is a function of the geometric design parameters, and then a numerical optimization procedure is used to find the parameters that minimize the cost function (hence, this approach leads to what is called an *optimal design*) [18.143–145]. There are, however, numerous drawbacks to this approach: the determination and effect of the weights in the cost function are difficult to ascertain, imperative requirements are difficult to incorporate and complexify the optimization process, and the definition of the performance indices is not trivial, only to name a few. The main issues can be stated as follows:

- It is quite difficult to ensure that the global minima is reached in spite of using relatively new optimization methods such as neural network or genetic algorithm.
- The robustness of the design solution obtained with the cost function approach with respect to the uncertainties in the final design. Indeed, the real instantiation of a theoretical solution always differs from the theoretical solution itself because of the manufacturing tolerances and other uncertainties that are inherent to a mechanical system.
- Performance requirements may be antagonistic (e.g., workspace and accuracy), and the approach only provides a compromise between these requirements that is difficult to master through the weights.

An alternative to optimal design is referred to as *appropriate design*, in which no optimization is considered, but the objective is to ensure that desired requirements are satisfied. This approach is based on the definition of the *parameter space* in which each dimension is associated with a design parameter. Performance requirements are considered in turn and the regions of the parameter space which correspond to robots satisfying the requirements are calculated. The design solution is then obtained as the intersection of the regions obtained for each individual requirement.

In practice, only an approximation of the regions is necessary since values close to the boundary cannot be physically realized due to the manufacturing tolerances. For that calculation, interval analysis was successfully used in various applications [18.36, 146].

The appropriate design approach is clearly more complex to implement than the cost function approach but has the advantage of providing all design solutions, with the consequence that manufacturing tolerances may be taken into account to ensure that the real robot also satisfies the desired requirements.

18.10 Wire-Driven Parallel Robots

Parallel mechanisms naturally lead themselves to another paradigm that exploits the structural properties of wires (cables). Cables are mechanical components that can withstand large tensile loads relative to their weight. When applied to parallel mechanisms, a plurality of cables can be attached to a moving platform and driven by actuated reels mounted on the base frame. Because of the parallel architecture, it is possible to ensure that cables can be maintained under tension within a given workspace. Wire-driven parallel mechanisms have been studied by several researchers over the last 20 years (e.g., [18.8, 147] for early work). The advantages of these mechanisms are numerous. When wound on a spool, cables allow motion ranges that are much larger than that produced by conventional articulated systems. Also, because they can only resist to tensile forces, cables are, as mentioned above, much thinner and lighter than most conventional mechanical components, for similar payloads. Thus, they have very low inertia and are particularly suitable for systems in which large ranges of motion are required. Indeed, long-reach articulated robots are heavy and involve the motion of significant mass, whereas cable-driven robots remain light even when the range of motion is very large. This property is clearly exemplified by aerial video camera systems such as the SkyCam [18.148] or in VIDEO 44, which shows an application in the scanning of artifacts for the generation of digital three-dimensional (3-D) models. VIDEO 45 and VIDEO 50 show another application in material handling.

Because wires can only resist tensile loads, wire-driven parallel mechanisms are generally designed based on either one of the two following approaches, namely: (1) the weight (gravity) of the moving platform is used to ensure that all cables are under tension or (2) the number of cables is larger than the number of degrees of freedom of the platform, thereby allowing

Avidbots Ex1016
Page 494 of 2154

a pre-stressing of the cables that can be used to ensure that all cables remain under tension. The wire-driven parallel mechanisms that are designed based on the first approach are referred to as *cable-suspended parallel mechanisms*, while the mechanisms that are based on the second approach are referred to as *fully constrained wire-driven parallel mechanisms*.

18.10.1 Cable-Suspended Parallel Mechanisms

Most cable-suspended parallel mechanisms are not redundantly actuated: they typically include as many actuators as DOFs. These mechanisms have been proposed in the literature as potential candidates for applications that require very large workspaces or mechanisms that can provide effective payload to mass ratios. One of the first cable-suspended mechanisms that was built is the Robocrane [18.8], developed by NIST almost two decades ago. This 6-DOF robot was intended for crane-type operations in which the pose of the payload can be fully controlled. Other cable-suspended mechanisms have also been studied and prototypes were built to validate their performance (e.g., [18.149] and [18.150]). More recently, cable-suspended parallel mechanisms including more actuators than DOFs were proposed, and it was shown that their workspace-to-footprint ratio can be much larger than that of nonredundant cable-suspended parallel mechanisms [18.151]. This is convincingly illustrated in VIDEO 45 and VIDEO 50, where a cable-suspended parallel mechanism with eight cables is demonstrated.

Most of the cable-suspended parallel mechanisms proposed or built in the past were assumed to work under static or quasi-static conditions. Under this assumption, the workspace can be determined based on the static equilibrium of the moving platform. The workspace is defined as the set of platform poses for which static equilibrium can be obtained while maintaining tension in all cables. Techniques to determine the static workspace of cable-suspended robots were proposed in the literature, for instance, in [18.152]. The dynamic trajectory planning and control of cable-suspended parallel mechanisms were also investigated in the recent literature (e.g., [18.153–156]), leading to cable-suspended parallel mechanisms that can be operated beyond their static workspace. The concept of dynamic workspace then arises [18.157], which can be defined as the set of poses that the platform can reach with at least one kinematic state (position, velocity, and acceleration).

Finally, it should be mentioned that the determination of the static pose of an underactuated cable-suspended parallel mechanism – that is, one with fewer cables than DOFs – is a very complex problem that has not yet been completely solved [18.158, 159]. In fact, for a given platform and given cable lengths, it is possible that an equilibrium pose can be reached, although some of the cables are not under tension. Therefore, the determination of the static pose involves not only solving for the actual number of cables, n, but also for all subrobots with $n-1$ cables.

18.10.2 Fully Constrained Wire-Driven Parallel Mechanisms

Fully constrained wire-driven parallel mechanisms have also attracted a great deal of attention. Fully constrained mechanisms include a number of cables larger than the number of degrees of freedom of the mechanism. If a proper geometry is used, they generally exhibit a wrench-closed workspace within which the platform can resist arbitrary external wrenches, while maintaining the cables in tension.

The statics of wire-driven robots can be described by

$$\mathbf{W}\boldsymbol{\tau} = \boldsymbol{f}\,, \tag{18.13}$$

where $\boldsymbol{\tau}$ is the vector of tensions in the wires, $\boldsymbol{f}$ is the wrench at the platform, and $\mathbf{W}$ is the so-called wrench matrix, which is configuration dependent. Since fully constrained wire-driven parallel mechanisms include more cables than DOFs, their wrench matrix $\mathbf{W}$ contains more columns than rows. Based on the above equation, the wrench-closed workspace can then be defined as the set of poses for which there exists at least one vector in the nullspace of $\mathbf{W}$ that has all its components of the same sign. When this condition is satisfied, the mechanism is said to be tensionable [18.160] and can theoretically resist arbitrary external wrenches simply by increasing the pre-stress in the cables.

The determination and the optimization of the wrench-closure workspace is a challenging problem that has attracted significant attention. Although the early work presented in [18.149, 161, 162] was mainly applicable to planar wire-driven robots, it paved the way to more general analyses by providing a mathematical modelling of the problem. Determining whether a given pose of the platform is located within the wrench-closure workspace is a rather straightforward problem. However, discretizing the workspace and applying such a criterion to a mesh of points does not solve the problem satisfactorily because the wrench-closed workspace is generally not convex and may even be composed of disconnected components. Therefore, more powerful techniques are needed in order to obtain reliable and useful results. In [18.163], in-

terval analysis is used to determine the wrench-closed workspace of fully constrained wire-driven parallel robots while providing definite answers. VIDEO 45 and VIDEO 50 show an example of fully constrained wire-driven parallel mechanisms.

18.10.3 Design and Prototyping

Synthesizing a wire-driven parallel mechanism for a prescribed wrench-closed workspace is a very challenging issue [18.164]. Also, in addition to the workspace, several other issues must be considered in the design of wire-driven parallel robots, for example, the determination of the potential interferences between the wires. An elegant geometric solution to this problem is proposed in [18.165] but handling translations and rotations simultaneously remains a challenge. Other design problems include the handling of special configurations and assembly mode changes as well as the accuracy and the practical implementation of the spools [18.166].

Several prototypes of wire-driven parallel robots were built and reported in the literature. Early prototypes include the NIST Robocrane [18.8], the Falcon robot [18.147], and the skycam system [18.148]. Many others were built, including those shown in the videos accompanying this chapter and the potential applications are numerous.

18.11 Application Examples

Parallel robots have been successfully used in many applications, a few of which are now briefly mentioned. Almost all recent land-based telescopes use parallel robots, either as a secondary mirror alignment system (e.g., the University of Arizona MMT or the ESO VISTA) or as a primary mirror pointing device [18.6]. A wire-driven parallel robot flew in the space shuttle mission STS-63 in February 1999 while an octopod (a parallel robot with eight legs) was used to isolate the space shuttle payload from vibration. All flight simulators use a parallel structure as a motion platform. They are nowadays used for driving simulators as well (e.g., the NADS-driving simulator [18.167]). In industry, numerous machine tools based on parallel structures have been designed. Some of them have found a niche market (e.g., the Tricept) and it can be expected that more will be used in the future. Ultra accurate positioning devices based on parallel robots are proposed by companies, such as Physik Instrumente, Alio, Micos, and Symétrie. In the food industry, the Delta robot proposed by ABB and Demaurex is widely used for fast packaging. Other companies such as Adept are also developing fast parallel robots with 3- and 4-DOFs. Medical applications are also growing, for example, in rehabilitation (after the pioneering work of the Rutgers ankle [18.168] and the platform Caren of Motek), in surgery (for example for spine surgery with the Renaissance robot of Mazor or for vitreoretinal surgery [18.169]) or for training and monitoring the motion of human joints [18.170], a topic in which cable-driven robots play an important role, as they do for aerial robots [18.171] or rescue operation (a version of the SkyCam robot has been used to fight oil fires during the Iraqi War).

18.12 Conclusion and Further Reading

The analysis of a parallel mechanism may be partly based on methods that are presented in other chapters of this handbook:

- *Kinematics*: Kinematics background is covered in Chap. 2.
- *Dynamics*: General approaches are presented in Chap. 3 while identification of the dynamic parameters is addressed in Chap. 6.
- *Design*: Design methodologies are covered in Chap. 16.
- *Control*: Control issues are presented in Chaps. 7, 8, and 9 although the closed-loop structure of parallel robots may require some adaptation of the control schemes.

It must also be emphasized that efficient numerical analysis is a key point for many algorithms related to parallel robots. A system solving with Gröbner basis, continuation method, and interval analysis is essential for kinematics, workspace, and singularity analysis. Further information and up-to-date extensive references and papers related to parallel robots can be found on the following two websites [18.38, 100]. Useful complementary readings on parallel robots are [18.140, 147, 164]. Parallel robots are slowly finding their way into various applications, not only in industry but also in field and service robotics, as discussed in Part F of this handbook. Still, compared to their serial counterpart, their analysis is far from being complete.

Video-References

VIDEO 43 3-DOF high-speed 3-RPS parallel robot
available from http://handbookofrobotics.org/view-chapter/18/videodetails/43
VIDEO 44 6-DOF cable-suspended robot
available from http://handbookofrobotics.org/view-chapter/18/videodetails/44
VIDEO 45 CoGiRo
available from http://handbookofrobotics.org/view-chapter/18/videodetails/45
VIDEO 46 Parallel 5R robot
available from http://handbookofrobotics.org/view-chapter/18/videodetails/46
VIDEO 47 Diamond
available from http://handbookofrobotics.org/view-chapter/18/videodetails/47
VIDEO 48 6-DOF statically balanced parallel robot
available from http://handbookofrobotics.org/view-chapter/18/videodetails/48
VIDEO 49 3-DOF dynamically balanced parallel robot
available from http://handbookofrobotics.org/view-chapter/18/videodetails/49
VIDEO 50 IPAnema
available from http://handbookofrobotics.org/view-chapter/18/videodetails/50
VIDEO 51 Par2 robot
available from http://handbookofrobotics.org/view-chapter/18/videodetails/51
VIDEO 52 Quadrupteron robot
available from http://handbookofrobotics.org/view-chapter/18/videodetails/52
VIDEO 53 R4 robot
available from http://handbookofrobotics.org/view-chapter/18/videodetails/53
VIDEO 54 Tripteron robot
available from http://handbookofrobotics.org/view-chapter/18/videodetails/54

References

18.1 J.E. Gwinnett: Amusement device, US Patent 178 9680 (1931)
18.2 V.E. Gough: Contribution to discussion of papers on research in automobile stability, control and tyre performance, 1956–1957. Proc. Auto Div. Inst. Mech. Eng. (1956)
18.3 D. Stewart: A platform with 6 degrees of freedom, Proc. Inst. Mech. Eng. Part 1 15 (1965) pp. 371–386
18.4 I.A. Bonev: The true origins of parallel robots, http://www.parallemic.org/Reviews/Review007.html (2003)
18.5 S. Briot, I.A. Bonev: Are parallel robots more accurate than serial robots?, CSME Transactions **31**(4), 445–456 (2007)
18.6 G. Aridon, D. Rémond, F. Morestin, L. Blanchard, R. Dufour: Self-deployment of a tape-spring hexapod: experimental and numerical investigation, ASME J. Mech. Des. **131**(2), 0210031–0210037 (2009)
18.7 M. Hafez, M.D. Lichter, S. Dubowsky: Optimized binary modular reconfigurable robotic devices, IEEE Trans. Mechatron. **8**(1), 152–162 (2003)
18.8 J. Albus, R. Bostelman, N. Dagalakis: The NIST robocrane, J. Robotic Syst. **10**(5), 709–724 (1993)
18.9 R. Clavel: DELTA, a fast robot with parallel geometry, 18th Int. Symp. Ind. Robots (ISIR), Lausanne (1988) pp. 91–100
18.10 K.H. Hunt: Structural kinematics of in parallel actuated robot arms, J. Mech. Transm. Autom. Des. **105**(4), 705–712 (1983)
18.11 K.E. Neumann: Robot: Neos Product HB Norrtalje Suède. US Patent 4732525, (1988)
18.12 Q. Li, J.M. Hervè: Parallel mechanisms with bifurcation of Schoenflies motion, IEEE Trans. Robotics **25**(1), 158–164 (2009)
18.13 J.M. Hervé: Group mathematics and parallel link mechanisms, 9th IFToMM World Congr. Theory Mach. Mech., Milan (1995) pp. 2079–2082
18.14 C.-C. Lee, J.M. Hervè: Type synthesis of primitive Schönflies-motion generators, Mech. Mach. Theory **44**(10), 1980–1997 (2009)
18.15 J. Meng, G.F. Liu, Z. Li: A geometric theory for analysis and synthesis of sub-6 DOF parallel manipulators, IEEE Trans. Robotics **23**(4), 625–649 (2007)
18.16 X. Kong, C.M. Gosselin: *Type Synthesis of Parallel Mechanisms*, Springer Tracts in Advanced Robotics, Vol. 33 (Springer, Berlin, Heidelberg 2007)
18.17 J.P. Merlet: http://www.sop.inria.fr/members/Jean-Pierre.Merlet//Archi/archi_robot.html (INRIA, France)
18.18 F. Pierrot, V. Nabat, O. Company, S. Krut, P. Poignet: Optimal design of a 4-DOF parallel manipulator: From academia to industry, IEEE Trans. Robotics **25**(2), 213–224 (2009)
18.19 C. Gosselin, T. Lalibertè, A. Veillette: Singularity-free kinematically redundant planar parallel mechanisms with unlimited rotational capability, IEEE Trans. Robotics **31**(2), 457–467 (2015)

Avidbots Ex1016
Page 497 of 2154

18.20 G. Gogu: *Structural Synthesis of Parallel Robots* (Kluwer, Dordrecht 2007)

18.21 P. Dietmaier: The Stewart Gough platform of general geometry can have 40 real postures. In: *Advances in Robot Kinematics: Analysis Control*, ed. by J. Lenarčič, M.L. Husty (Kluwer, Dordrecht 1998) pp. 7–16

18.22 M.L. Husty: An algorithm for solving the direct kinematic of Stewart Gough-type platforms, Mech. Mach. Theory **31**(4), 365–380 (1996)

18.23 M. Raghavan: The Stewart platform of general geometry has 40 configurations, ASME J. Mech. Des. **115**(2), 277–282 (1993)

18.24 J.C. Faugère, D. Lazard: The combinatorial classes of parallel manipulators, Mech. Mach. Theory **30**(6), 765–776 (1995)

18.25 T.-Y. Lee, J.-K. Shim: Improved dyalitic elimination algorithm for the forward kinematics of the general Stewart Gough platform, Mech. Mach. Theory **38**(6), 563–577 (2003)

18.26 F. Rouillier: Real roots counting for some robotics problems. In: *Computational Kinematics*, ed. by J.-P. Merlet, B. Ravani (Kluwer, Dordrecht 1995) pp. 73–82

18.27 J.-P. Merlet: Solving the forward kinematics of a Gough-type parallel manipulator with interval analysis, Int. J. Robotics Res. **23**(3), 221–236 (2004)

18.28 D. Manocha: Algebraic and Numeric Techniques for Modeling and Robotics, Ph.D. Thesis (University of California, Berkeley 1992)

18.29 A.J. Sommese, C.W. Wampler: *The Numerical Solution of Systems of Polynomials: Arising in Engineering And Science* (World Scientific, Singapore 2005)

18.30 C. Gosselin, J. Sefrioui, M.J. Richard: Solution polynomiale au problème de la cinématique directe des manipulateurs parallèles plans à 3 degrés de liberté, Mech. Mach. Theory **27**(2), 107–119 (1992)

18.31 P. Wenger, D. Chablat, M. Zein: Degeneracy study of the forward kinematics of planar 3 — *RPR* parallel manipulators, ASME J. Mech. Des. **129**(12), 1265–1268 (2007)

18.32 L. Baron, J. Angeles: The direct kinematics of parallel manipulators under joint-sensor redundancy, IEEE Trans. Robotics Autom. **16**(1), 12–19 (2000)

18.33 J.-P. Merlet: Closed-form resolution of the direct kinematics of parallel manipulators using extra sensors data, IEEE Int. Conf. Robotics Autom. (ICRA), Atlanta (1993) pp. 200–204

18.34 R. Vertechy, V. Parenti-Castelli: Robust, fast and accurate solution of the direct position analysis of parallel manipulators by using extra-sensors. In: *Parallel Manipulators: Towards New Applications*, ed. by H. Wu (InTech, Rijeka 2008) pp. 133–154

18.35 R. Stoughton, T. Arai: Kinematic optimization of a chopsticks-type micro-manipulator, Jpn. Symp. Flex. Autom., San Fransisco (1993) pp. 151–157

18.36 J.-P. Merlet, D. Daney: Dimensional synthesis of parallel robots with a guaranteed given accuracy over a specific workspace, IEEE Int. Conf. Robotics Autom. (ICRA), Barcelona (2005)

18.37 J.-P. Merlet: Jacobian, manipulability, condition number, and accuracy of parallel robots, ASME J. Mech. Des. **128**(1), 199–206 (2006)

18.38 F.-T. Niaritsiry, N. Fazenda, R. Clavel: Study of the source of inaccuracy of a 3 DOF flexure hinge-based parallel manipulator, IEEE Int. Conf. Robotics Autom. (ICRA), New Orleans (2004) pp. 4091–4096

18.39 G. Pritschow, C. Eppler, T. Garber: Influence of the dynamic stiffness on the accuracy of PKM, 3rd Chemnitzer Parallelkinematik Semin. Chemnitz (2002) pp. 313–333

18.40 V. Parenti-Castelli, S. Venanzi: On the joint clearance effects in serial and parallel manipulators, Workshop Fundam. Issues Future Res. Dir. Parallel Mech. Manip. Québec (2002) pp. 215–223

18.41 A. Pott, M. Hiller: A new approach to error analysis in parallel kinematic structures, Adv. Robot Kinemat., Sestri-Levante (2004)

18.42 K. Wohlhart: Degrees of shakiness, Mech. Mach. Theory **34**(7), 1103–1126 (1999)

18.43 K. Tönshoff, B. Denkena, G. Günther, H.C. Möhring: Modelling of error effects on the new hybrid kinematic DUMBO structure, 3rd Chemnitzer Parallelkinematik Semin. Chemnitz (2002) pp. 639–653

18.44 U. Sellgren: Modeling of mechanical interfaces in a systems context, Int. ANSYS Conf., Pittsburgh (2002)

18.45 S. Eastwood, P. Webbb: Compensation of thermal deformation of a hybrid parallel kinematic machine, Robotics Comput. Manuf. **25**(1), 81–90 (2009)

18.46 W. Khalil, S. Besnard: Identificable parameters for the geometric calibration of parallel robots, Arch. Control Sci. **11**(3/4), 263–277 (2001)

18.47 C.W. Wampler, J.M. Hollerbach, T. Arai: An implicit loop method for kinematic calibration and its application to closed-chain mechanisms, IEEE Trans. Robotics Autom. **11**(5), 710–724 (1995)

18.48 D. Daney, Y. Papegay, A. Neumaier: Interval methods for certification of the kinematic calibration of parallel robots, IEEE Int. Conf. Robotics Autom. (ICRA), New Orleans (2004) pp. 1913–1918

18.49 D. Daney, N. Andreff, G. Chabert, Y. Papegay: Interval method for calibration of parallel robots: a vision-based experimentation, Mech. Mach. Theory **41**(8), 929–944 (2006)

18.50 A. Nahvi, J.M. Hollerbach: The noise amplification index for optimal pose selection in robot calibration, IEEE Int. Conf. Robotics Autom. (ICRA), Minneapolis (1996) pp. 647–654

18.51 G. Meng, L. Tiemin, Y. Wensheng: Calibration method and experiment of Stewart platform using a laser tracker, Int. Conf Syst. Man Cybern., Hague (2003) pp. 2797–2802

Avidbots Ex1016
Page 498 of 2154

18.52 D. Daney: Optimal measurement configurations for Gough platform calibration, IEEE Int. Conf. Robotics Autom. (ICRA), Washington (2002) pp. 147–152

18.53 P. Renaud, N. Andreff, P. Martinet, G. Gogu: Kinematic calibration of parallel mechanisms: A novel approach using legs observation, IEEE Trans. Robotics **21**(4), 529–538 (2005)

18.54 C. Gosselin, J. Angeles: Singularity analysis of closed-loop kinematic chains, IEEE Trans. Robotics Autom. **6**(3), 281–290 (1990)

18.55 D. Zlatanov, R.G. Fenton, B. Benhabib: A unifying framework for classification and interpretation of mechanism singularities, ASME J. Mech. Des. **117**(4), 566–572 (1995)

18.56 D. Zlatanov, I.A. Bonev, C.M. Gosselin: Constraint singularities of parallel mechanisms, IEEE Int. Conf. Robotics Autom. (ICRA), Washington (2002) pp. 496–502

18.57 M. Conconi, M. Carricato: A new assessment of singularities of parallel kinematic chains, IEEE Trans. Robotics **25**(4), 757–770 (2009)

18.58 G. Liu, Y. Lou, Z. Li: Singularities of parallel manipulators: A geometric treatment, IEEE Trans. Robotics Autom. **19**(4), 579–594 (2003)

18.59 I.A. Bonev, D. Zlatanov: The mystery of the singular SNUtranslational parallel robot, http://www.parallemic.org/Reviews/Review004.html (2001)

18.60 H. Li, C.M. Gosselin, M.J. Richard, B. Mayer St-Onge: Analytic form of the six-dimensional singularity locus of the general Gough-Stewart platform, ASME J. Mech. Des. **128**(1), 279–287 (2006)

18.61 B. Mayer St-Onge, C.M. Gosselin: Singularity analysis and representation of the general Gough-Stewart platform, Int. J. Robotics Res. **19**(3), 271–288 (2000)

18.62 J.-P. Merlet: Singular configurations of parallel manipulators and Grassmann geometry, Int. J. Robotics Res. **8**(5), 45–56 (1989)

18.63 R. Ben-Horin, M. Shoham: Application of Grassmann Cayley algebra to geometrical interpretation of parallel robot singularities, Int. J. Robotics Res. **28**(1), 127–141 (2009)

18.64 H. Pottmann, M. Peternell, B. Ravani: Approximation in line space. Applications in robot kinematics, Adv. Robot Kinemat., Strobl (1998) pp. 403–412

18.65 P.A. Voglewede, I. Ebert-Uphoff: Measuring *closeness* to singularities for parallel manipulators, IEEE Int. Conf. Robotics Autom. (ICRA), New Orleans (2004) pp. 4539–4544

18.66 G. Nawratil: New performance indices for 6-DOF UPS and 3-DOF RPR parallel manipulators, Mech. Mach. Theory **44**(1), 208–221 (2009)

18.67 J. Hubert, J.-P. Merlet: Static of parallel manipulators and closeness to singularity, J. Mech. Robotics **1**(1), 1–6 (2009)

18.68 P. Cardou, S. Bouchard, C. Gosselin: Kinematic-sensitivity indices for dimensionally nonhomogeneous jacobian matrices, IEEE Trans. Robotics **26**(1), 166–173 (2010)

18.69 J.-P. Merlet, D. Daney: A formal-numerical approach to determine the presence of singularity within the workspace of a parallel robot. In: *Computational Kinematics*, ed. by F.C. Park, C.C. Iurascu (EJCK, Seoul 2001) pp. 167–176

18.70 S. Bhattacharya, H. Hatwal, A. Ghosh: Comparison of an exact and an approximate method of singularity avoidance in platform type parallel manipulators, Mech. Mach. Theory **33**(7), 965–974 (1998)

18.71 D.N. Nenchev, M. Uchiyama: Singularity-consistent path planning and control of parallel robot motion through instantaneous-self-motion type, IEEE Int. Conf. Robotics Autom. (ICRA), Minneapolis (1996) pp. 1864–1870

18.72 C. Innocenti, V. Parenti-Castelli: Singularity-free evolution from one configuration to another in serial and fully-parallel manipulators, 22nd Bienn. Mech. Conf., Scottsdale (1992) pp. 553–560

18.73 M. Husty: Non-singular assembly mode change in 3-RPR parallel manipulators. In: *Computational Kinematics*, ed. by A. Kecskeméthy, A. Müller (Springer, Berlin, Heidelberg 2009) pp. 51–60

18.74 S. Caro, P. Wenger, D. Chablat: Non-singular assembly mode changing trajectories of a 6-DOF parallel robot, ASME Int. Des. Eng. Techn. Conf. Comput. Inform. Eng. Conf., Chicago (2012)

18.75 R. Ranganath, P.S. Nair, T.S. Mruthyunjaya, A. Ghosal: A force-torque sensor based on a Stewart platform in a near-singular configuration, Mech. Mach. Theory **39**(9), 971–998 (2004)

18.76 M.L. Husty, A. Karger: Architecture singular parallel manipulators and their self-motions. In: *Adv. Robot Kinematics: Analysis Control*, ed. by J. Lenarčič, M.L. Husty (Springer, Dordrecht 2000) pp. 355–364

18.77 A. Karger: Architecture singular planar parallel manipulators, Mech. Mach. Theory **38**(11), 1149–1164 (2003)

18.78 K. Wohlhart: Mobile 6-SPS parallel manipulators, J. Robotic Syst. **20**(8), 509–516 (2003)

18.79 C. Innocenti, V. Parenti-Castelli: Direct kinematics of the 6-4 fully parallel manipulator with position and orientation uncoupled, Eur. Robotics Intell. Syst. Conf., Corfou (1991)

18.80 G. Gogu: Mobility of mechanisms: A critical review, Mech. Mach. Theory **40**(10), 1068–1097 (2005)

18.81 I. Zabalza, J. Ros, J.J. Gil, J.M. Pintor, J.M. Jimenez: Tri-Scott a new kinematic structure for a 6-DOF decoupled parallel manipulator, Workshop Fundam. Issues Future Res. Dir. Parallel Mech. Manip., Québec (2002) pp. 12–15

18.82 C. Gosselin: Determination of the workspace of 6-DOF parallel manipulators, ASME J. Mech. Des. **11**(3), 331–336 (1990)

18.83 J.-P. Merlet: Geometrical determination of the workspace of a constrained parallel manipulator, Adv. Robot Kinemat., Ferrare (1992) pp. 326–329

18.84 F.A. Adkins, E.J. Haug: Operational envelope of a spatial Stewart platform, ASME J. Mech. Des. **119**(2), 330–332 (1997)

Avidbots Ex1016
Page 499 of 2154

18.85 E.J. Haugh, F.A. Adkins, C.M. Luh: Operational envelopes for working bodies of mechanisms and manipulators, ASME J. Mech. Des. **120**(1), 84–91 (1998)

18.86 O. Bohigas, L. Ros, M. Manubens: A complete method for workspace boundary determination, Adv. Robot Kinemat., Piran (2010) pp. 329–338

18.87 J.-P. Merlet: Determination of 6D workspaces of Gough-type parallel manipulator and comparison between different geometries, Int. J. Robotics Res. **18**(9), 902–916 (1999)

18.88 P. Wenger, D. Chablat: Workspace and assembly modes in fully parallel manipulators: A descriptive study. In: *Advances in Robot Kinematics: Analysis Control*, ed. by J. Lenarčič, M.L. Husty (Springer, Dordrecht 1998) pp. 117–126

18.89 G. Moroz, F. Rouiller, D. Chablat, P. Wenger: On the determination of cusp points of 3-*RPR* parallel manipulators, Mech. Mach. Theory **45**(11), 1555–1567 (2010)

18.90 S. Briot, V. Arakelian: Optimal force generation in parallel manipulators for passing through the singular positions, Int. J. Robotics Res. **27**(2), 967–983 (2008)

18.91 J. Hesselbach, C. Bier, C. Budde, P. Last, J. Maaß, M. Bruhn: Parallel robot specific control fonctionalities, 2nd Int. Colloquium, Collab. Res. Centre, Braunschweig (2005) pp. 93–108

18.92 J.-P. Merlet: An efficient trajectory verifier for motion planning of parallel machine, Parallel Kinemat. Mach. Int. Conf., Ann Arbor (2000)

18.93 R. Ur-Rehman, S. Caro, D. Chablat, P. Wenger: Multi-objective path placement of parallel kinematics machines based on energy consumption, shaking forces and maximum actuator torques: Application to the Orthoglide, Mech. Mach. Theory **45**(8), 1125–1141 (2010)

18.94 J. Cortés, T. Siméon: Probabilistic motion planning for parallel mechanisms, IEEE Int. Conf. Robotics Autom. (ICRA), Taipei (2003) pp. 4354–4359

18.95 J.H. Yakey, S.M. LaValle, L.E. Kavraki: Randomized path planning for linkages with closed kinematic chains, IEEE Trans. Robotics Autom. **17**(6), 951–958 (2001)

18.96 J.-P. Merlet, M.-W. Perng, D. Daney: Optimal trajectory planning of a 5-axis machine tool based on a 6-axis parallel manipulator. In: *Advances in Robot Kinematics*, ed. by J. Lenarčič, M.L. Husty (Kluwer, Dordrecht 2000) pp. 315–322

18.97 D. Shaw, Y.-S. Chen: Cutting path generation of the Stewart platform-based milling machine using an end-mill, Int. J. Prod. Res. **39**(7), 1367–1383 (2001)

18.98 Z. Wang, Z. Wang, W. Liu, Y. Lei: A study on workspace, boundary workspace analysis and workpiece positioning for parallel machine tools, Mech. Mach. Theory **36**(6), 605–622 (2001)

18.99 D.R. Kerr: Analysis, properties, and design of a Stewart-platform transducer, J. Mech. Transm. Autom. Des. **111**(1), 25–28 (1989)

18.100 C.C. Nguyen, S.S. Antrazi, Z.L. Zhou, C.E. Campbell: Analysis and experimentation of a Stewart platform-based force/torque sensor, Int. J. Robotics Autom. **7**(3), 133–141 (1992)

18.101 C. Reboulet, A. Robert: Hybrid control of a manipulator with an active compliant wrist, 3rd Int. Symp. Robotics Res., Gouvieux (1985) pp. 76–80

18.102 J. Duffy: *Statics and Kinematics with Applications to Robotics* (Cambridge Univ. Press, New York 1996)

18.103 C. Huang, W.-H. Hung, I. Kao: New conservative stiffness mapping for the Stewart Gough platform, IEEE Int. Conf. Robotics Autom. (ICRA), Washington (2002) pp. 823–828

18.104 J.L. Herder: Energy-free systems: Theory, Conception and Design of Statically Balanced Spring Mechanisms, Ph.D. Thesis (Delft University of Technology, Delft 2001)

18.105 G.R. Dunlop, T.P. Jones: Gravity counter balancing of a parallel robot for antenna aiming, 6th Int. Symp. Robotics Manuf., Montpellier (1996) pp. 153–158

18.106 M. Jean, C. Gosselin: Static balancing of planar parallel manipulators, IEEE Int. Conf. Robotics Autom. (ICRA), Minneapolis (1996) pp. 3732–3737

18.107 I. Ebert-Uphoff, C.M. Gosselin, T. Laliberté: Static balancing of spatial parallel platform-revisited, ASME J. Mech. Des. **122**(1), 43–51 (2000)

18.108 C.M. Gosselin, J. Wang: Static balancing of spatial six-degree-of-freedom parallel mechanisms with revolute actuators, J. Robotic Syst. **17**(3), 159–170 (2000)

18.109 M. Leblond, C.M. Gosselin: Static balancing of spatial and planar parallel manipulators with prismatic actuators, ASME Des. Eng. Tech. Conf., Atlanta (1998) pp. 5187–5193

18.110 B. Monsarrat, C.M. Gosselin: Workspace analysis and optimal design of a 3-leg 6-DOF parallel platform mechanism, IEEE Trans. Robotics Autom. **19**(6), 954–966 (2003)

18.111 J. Wang, C.M. Gosselin: Static balancing of spatial three-degree-of-freedom parallel mechanisms, Mech. Mach. Theory **34**(3), 437–452 (1999)

18.112 Y. Wu, C.M. Gosselin: Synthesis of reactionless spatial 3-DOF and 6-DOF mechanisms without separate counter-rotations, Int. J. Robotics Res. **23**(6), 625–642 (2004)

18.113 M. Ait-Ahmed: Contribution à la Modélisation Géométrique et Dynamique des Robots Parallèles, Ph.D. Thesis (Univ. Paul Sabatier, Toulouse 1993)

18.114 G.F. Liu, X.Z. Wu, Z.X. Li: Inertial equivalence principle and adaptive control of redundant parallel manipulators, IEEE Int. Conf. Robotics Autom. (ICRA), Washington (2002) pp. 835–840

18.115 R. Clavel: Conception d'un robot parallèle rapide à 4 degrés de liberté, Ph.D. Thesis (EPFL, Lausanne, 1991), No. 925.

18.116 J. Gallardo, J.M. Rico, A. Frisoli, D. Checcacci, M. Bergamasco: Dynamics of parallel manipulators by means of screw theory, Mech. Mach. Theory **38**(11), 1113–1131 (2003)

18.117 L.-W. Tsai: Solving the inverse dynamics of a Stewart Gough manipulator by the principle

Avidbots Ex1016
Page 500 of 2154

of virtual work, ASME J. Mech. Des. **122**(1), 3–9 (2000)
18.118 J. Wang, C.M. Gosselin: A new approach for the dynamic analysis of parallel manipulators, Multibody Syst. Dyn. **2**(3), 317–334 (1998)
18.119 Z. Geng, L.S. Haynes: On the dynamic model and kinematic analysis of a class of Stewart platforms, Robotics Auton. Syst. **9**(4), 237–254 (1992)
18.120 K. Liu, F. Lewis, G. Lebret, D. Taylor: The singularities and dynamics of a Stewart platform manipulator, J. Intell. Robotic Syst. **8**(3), 287–308 (1993)
18.121 K. Miller, R. Clavel: The Lagrange-based model of Delta-4 robot dynamics, Robotersysteme **8**(1), 49–54 (1992)
18.122 H. Abdellatif, B. Heimann: Computational efficient inverse dynamics of 6-DOF fully parallel manipulators by using the Lagrangian formalism, Mech. Mach. Theory **44**(1), 192–207 (2009)
18.123 K. Miller: Optimal design and modeling of spatial parallel manipulators, Int. J. Robotics Res. **23**(2), 127–140 (2004)
18.124 A. Codourey, E. Burdet: A body oriented method for finding a linear form of the dynamic equations of fully parallel robot, IEEE Int. Conf. Robotics Autom. (ICRA), Albuquerque (1997) pp. 1612–1618
18.125 B. Dasgupta, P. Choudhury: A general strategy based on the Newton Euler approach for the dynamic formulation of parallel manipulators, Mech. Mach. Theory **34**(6), 801–824 (1999)
18.126 P. Guglielmetti: Model-Based Control of Fast Parallel Robots: a Global Approach in Operational Space, Ph.D. Thesis (EPFL, Lausanne, 1994)
18.127 K. Harib, K. Srinivasan: Kinematic and dynamic analysis of Stewart platform-based machine tool structures, Robotica **21**(5), 541–554 (2003)
18.128 W. Khalil, O. Ibrahim: General solution for the dynamic modeling of parallel robots, IEEE Int. Conf. Robotics Autom. (ICRA), New Orleans (2004) pp. 3665–3670
18.129 C. Reboulet, T. Berthomieu: Dynamic model of a six degree of freedom parallel manipulator, Int. Conf. Adv. Robotics, Pise (1991) pp. 1153–1157
18.130 H. Abdellatif, B. Heimann: Experimental identification of the dynamics model for 6-DOF parallel manipulators, Robotica **28**(3), 359–368 (2010)
18.131 H. Abdellatif, B. Heimann: Model based control for industrial robots: Uniform approaches for serial and parallel structures. In: *Industrial Robotics: Theory, Modelling and Control*, ed. by S. Cubero (pro literatur Verlag, Augsburg 2007) pp. 523–556
18.132 S. Tadokoro: Control of parallel mechanisms, Adv. Robotics **8**(6), 559–571 (1994)
18.133 M. Honegger, A. Codourey, E. Burdet: Adaptive control of the Hexaglide, a 6 DOF parallel manipulator, IEEE Int. Conf. Robotics Autom. (ICRA), Albuquerque (1997) pp. 543–548
18.134 S. Bhattacharya, H. Hatwal, A. Ghosh: An online estimation scheme for generalized Stewart platform type parallel manipulators, Mech. Mach. Theory **32**(1), 79–89 (1997)
18.135 P. Guglielmetti, R. Longchamp: A closed-form inverse dynamics model of the Delta parallel robot, 4th IFAC Symp. Robot Control, Syroco, Capri (1994) pp. 51–56
18.136 K. Miller: Modeling of dynamics and model-based control of DELTA direct-drive parallel robot, J. Robotics Mechatron. **17**(4), 344–352 (1995)
18.137 E. Burdet, M. Honegger, A. Codourey: Controllers with desired dynamic compensation and their implementation on a 6 DOF parallel manipulator, IEEE Int. Conf. Intell. Robots Syst. (IROS), Takamatsu (2000)
18.138 K. Yamane, Y. Nakamura, M. Okada, N. Komine, K.I. Yoshimoto: Parallel dynamics computation and h_∞ acceleration control of parallel manipulators for acceleration display, ASME J. Dyn. Syst. Meas. Control **127**(2), 185–191 (2005)
18.139 J.E. McInroy: Modeling and design of flexure jointed Stewart platforms for control purposes, IEEE/ASME Trans. Mechatron. **7**(1), 95–99 (2002)
18.140 F. Paccot, N. Andreff, P. Martinet: A review on the dynamic control of parallel kinematic machines: theory and experiments, Int. J. Robotics Res. **28**(3), 395–416 (2009)
18.141 D. Corbel, M. Gouttefarde, O. Company, F. Pierrot: Towards 100G with PKM is actuation redundancy a good solution for pick-and-place?, IEEE Int. Conf. Robotics Autom. (ICRA), Anchorage (2010) pp. 4675–4682
18.142 F. Xi: Dynamic balancing of hexapods for high-speed applications, Robotica **17**(3), 335–342 (1999)
18.143 J. Angeles: The robust design of parallel manipulators, 1st Int. Colloquium, Collab. Res. Centre, Braunschweig (2002) pp. 9–30
18.144 S. Bhattacharya, H. Hatwal, A. Ghosh: On the optimum design of a Stewart platform type parallel manipulators, Robotica **13**(2), 133–140 (1995)
18.145 K.E. Zanganeh, J. Angeles: Kinematic isotropy and the optimum design of parallel manipulators, Int. J. Robotics Res. **16**(2), 185–197 (1997)
18.146 H. Fang, J.-P. Merlet: Multi-criteria optimal design of parallel manipulators based on interval analysis, Mech. Mach. Theory **40**(2), 151–171 (2005)
18.147 S. Kamamura, W. Choe, S. Tanaka, S.R. Pandian: Development of an Ultrahigh Speed Robot FALCON using Wire Drive System, IEEE Int. Conf. Robotics Autom. (ICRA) (1995) pp. 215–220
18.148 L.L. Cone: Skycam: An Aerial Robotic Camera System, Byte **10**(10), 122–132 (1985)
18.149 J. Pusey, A. Farrah, S.K. Agrawal, E. Messina: Design and workspace analysis of a 6-6 cable-suspended parallel robot, Mech. Mach. Theory **39**, 761–778 (2004)
18.150 C. Gosselin, S. Bouchard: A gravity-powered mechanism for extending the workspace of a cable-driven parallel mechanism: Application to the appearance modelling of objects, Int. J. Autom. Technol. **4**(4), 372–379 (2010)

Avidbots Ex1016
Page 501 of 2154

18.151 J. Lamaury, M. Gouttefarde: A tension distribution method with improved computational efficiency. In: *Cable-Driven Parallel Robots*, ed. by T. Bruckmann, A. Pott (Springer, Berlin, Heidelberg 2012) pp. 71–85

18.152 A.T. Riechel, I. Ebert-Uphoff: Force-feasible workspace analysis for underconstrained,point-mass cable robots, IEEE Int. Conf. Robotics Autom. (ICRA) (2004) pp. 4956–4962

18.153 D. Cunningham, H. Asada: The Winch-Bot: A cable-suspended, under-actuated robot utilizing parametric self-excitation, IEEE Int. Conf. Robotics Autom. (ICRA) (2009) pp. 1844–1850

18.154 S. Lefrançois, C. Gosselin: Point-to-point motion control of a pendulum-like 3-DOF underactuated cable-driven robot, IEEE Int. Conf. Robotics Autom. (ICRA) (2010) pp. 5187–5193

18.155 D. Zanotto, G. Rosati, S.K. Agrawal: Modeling and control of a 3-DOF pendulum-like manipulator, IEEE Int. Conf. Robotics Autom. (ICRA) (2011) pp. 3964–3969

18.156 C. Gosselin, P. Ren, S. Foucault: Dynamic trajectory planning of a two-DOF cable-suspended parallel robot, IEEE Int. Conf. Robotics Autom. (ICRA) (2012) pp. 1476–1481

18.157 G. Barrette, C. Gosselin: Determination of the dynamic workspace of cable-driven planar parallel mechanisms, ASME J. Mech. Des. **127**(2), 242–248 (2005)

18.158 M. Carricato, J.-P. Merlet: Direct geometrico-static problem of under-constrained cable-driven parallel robots with three cables, IEEE Int. Conf. Robotics Autom. (ICRA) (2011) pp. 3011–3017

18.159 J.-F. Collard, P. Cardou: Computing the lowest equilibrium pose of a cable-suspended rigid body, Optim. Eng. **14**, 457–476 (2013)

18.160 S. Behzadipour, A. Khajepour: Trajectory planning in cable-based high-speed parallel robots, IEEE Trans. Robotics **22**(3), 559–563 (2006)

18.161 M. Gouttefarde, C. Gosselin: Analysis of the wrench-closure workspace of planar parallel cable-driven mechanisms, IEEE Trans. Robotics **22**(3), 434–445 (2006)

18.162 E. Stump, V. Kumar: Workspace delienation of cable-actuated parallel manipulators, ASME Int. Des. Eng. Tech. Conf. (2004)

18.163 M. Gouttefarde, D. Daney, J.-P. Merlet: Interval-analysis-based determination of the wrench-feasible workspace of parallel cable-driven robots, IEEE Trans. Robotics **27**(1), 1–13 (2011)

18.164 K. Azizian, P. Cardou: The dimensional synthesis of planar parallel cable-driven mechanisms through convex relaxations, ASME J. Mech. Des. **4**(3), 0310111–03101113 (2012)

18.165 S. Perreault, P. Cardou, C. Gosselin, M.J.D. Otis: Geometric determination of the interference-free constant-orientation workspace of parallel cable-driven mechanisms, ASME J. Mech. Robotics **2**(3), 031016 (2010)

18.166 J.-P. Merlet: MARIONET, a family of modular wire-driven parallel robots, Adv. Robot Kinemat.: Motion Man Mach. (2010) pp. 53–61

18.167 The National Advanced Driving Simulator, The University of Iowa: http://www.nads-sc.uiowa.edu

18.168 M. Girone, G. Burdea, M. Bouzit, V. Popescu, J.E. Deutsch: A Stewart platform-based system for ankle telerehabilitation, Auton. Robots **10**(2), 203–212 (2001)

18.169 T. Nakano, M. Sugita, T. Ueta, Y. Tamaki, M. Mitsuishi: A parallel robot to assist vitreoretinal surgery, Int. J. Comput. Assist. Radiol. Surg. **4**(6), 517–526 (2009)

18.170 M. Wu, T.G. Hornby, J.M. Landry, H. Roth, B.D. Schmit: A cable-driven locomotor training system for restoration of gait in human SCI, Gait Posture **33**(2), 256–260 (2011)

18.171 J. Fink, N. Michael, S. Kim, V. Kumar: Planning and control for cooperative manipulation and transportation with aerial robots, Proc. Intl. Sym. Robot. Res., Luzern (2009)

Avidbots Ex1016
Page 502 of 2154

Multimedia Contents

19. Robot Hands

Claudio Melchiorri, Makoto Kaneko

Multifingered robot hands have a potential capability for achieving dexterous manipulation of objects by using rolling and sliding motions. This chapter addresses design, actuation, sensing and control of multifingered robot hands. From the design viewpoint, they have a strong constraint in actuator implementation due to the space limitation in each joint. After briefly introducing the overview of anthropomorphic end-effector and its dexterity in Sect. 19.1, various approaches for actuation are provided with their advantages and disadvantages in Sect. 19.2. The key classification is (1) remote actuation or build-in actuation and (2) the relationship between the number of joints and the number of actuator. In Sect. 19.3, actuators and sensors used for multifingered hands are described. In Sect. 19.4, modeling and control are introduced by considering both dynamic effects and friction. Applications and trends are given in Sect. 19.5. Finally, this chapter is closed with conclusions and further reading.

19.1 **Basic Concepts** 464
19.1.1 Anthropomorphic End-Effectors . 464
19.1.2 Dexterity of a Robotic Hand 465
19.2 **Design of Robot Hands** 465
19.2.1 Actuators Placement and Motion Transmission 465
19.2.2 Actuation Architectures 467
19.3 **Technologies for Actuation and Sensing** 470
19.3.1 Actuation 470
19.3.2 Sensors 470
19.4 **Modeling and Control of a Robot Hand** . 473
19.4.1 Dynamic Effects of Flexible Transmission Systems 474
19.4.2 Transmission Model of Tendon-Outer-Tube System 476
19.4.3 The Control Through Single-Acting Actuators 476
19.4.4 Control of a Robot Hand 477
19.5 **Applications and Trends** 477
19.6 **Conclusions and Further Reading** 478
Video-References 478
References 479

Human hands have great potentialities not only for grasping objects of various shapes and dimensions, but also for manipulating them in a dexterous manner. It is common experience that, by training, one can perform acrobatic manipulation of stick-shaped objects, manipulate a pencil by using rolling or sliding motions, perform precise operations requiring fine control of small tools or objects. It is obvious that this kind of dexterity cannot be achieved by a simple gripper capable of open/close motion only. A multifingered robot hand can therefore provide a great opportunity for achieving such a dexterous manipulation in a robotic system. Moreover, we have also to consider that human beings do not use hands only for grasping or manipulating objects. Exploration, touch, perception of physical properties (roughness, temperature, weight, just to mention a few) are other fundamental tasks that we usually are able to perform thanks to our hands. We expect this type of capabilities also from robotic end-effectors and therefore, by adding quite advanced sensing equipments and proper control strategies, we may improve the interaction capabilities with the environment, achieving for example active exploration, detection of sensing surface properties (local friction, impedance, and so on), tasks that are usually very hard or impossible for simple grippers. For these and other reasons the study of multifingered robot hands has strongly interested the research community since the early days of Robotics.

It was in late 1970s that Okada developed a multifingered robot hand with a tendon driving system

Avidbots Ex1016
Page 503 of 2154

and demonstrated a nut opening motion [19.1]. In early 1980s, two major projects on multifingered robot hands have been launched: the Stanford/Jet Propulsion Laboratory (JPL; VIDEO 751) hand and the Utah/Massachusetts Institute of Technology (MIT) hand [19.2, 3]. These two robot hands still represent a milestone and a term of comparison for the design of new devices. Since then, several multifingered hands have been designed and developed in a number research institutes all over the world. Among the most known, one can mention the Deutsches Zentrum für Luft- und Raumfahrt (DLR) hand(s) (VIDEO 754, VIDEO 768, and VIDEO 769), Mechanical Engineering Laboratory (MEL) hand, Electro-Technical Laboratory (ETL) hand, Darmstadt hand, Karlsruhe hand, University of Bologna (UB) hand (VIDEO 756, VIDEO 767), Barrett hand (VIDEO 752), Yasukawa hand, Gifu hand, U-Tokyo hand, Hiroshima hand, Soft Pisa/IIT hand (VIDEO 749, VIDEO 750), and many others [19.4–10].

When designing a multifingered hand, on the basis of its utilization, one should first define the following key issues: number and kinematic configuration of the fingers, anthropomorphic or nonanthropomorphic aspect, built-in or remote actuation, transmission system (in case of remote actuation), sensor assignment, integration with a carrying device (robot arm), control. All these aspects are considered in this chapter.

19.1 Basic Concepts

Before illustrating the main issues involved in the design and use of a robotic hand, it is necessary to discuss some basic concepts and definitions often encountered when dealing with these devices. In particular, terms like *dexterity* and *anthropomorphism* must be defined, and their implications on robotic hand design specified.

19.1.1 Anthropomorphic End-Effectors

The term *anthropomorphism* denotes the capability of a robotic end-effector to mimic the human hand, partly or totally, as far as shape, size, consistency, and general aspect (including color, temperature, and so on) are considered. As the word itself suggests, anthropomorphism is related to the external perceivable properties, and is not, itself, a measure of what the hand can do. On the contrary, *dexterity* is related to actual functionality and not to shape or aesthetic factors. In this sense anthropomorphism and dexterity are *orthogonal* concepts, whose reciprocal dependance (at least in the robotic field) has been not proved yet.

As a matter of fact, we can find in the literature anthropomorphic end-effectors with very poor dexterity level, even though they are called *hands*, as the tasks they can perform are limited to very rough grasping procedures [19.11]. Similarly, we can find smart end-effectors, capable of sophisticated manipulation procedures, without any level of anthropomorphism, e.g., the DxGrip-II [19.12]. Anthropomorphism itself is neither necessary nor sufficient to achieve dexterity, although it is quite evident that the human hand achieves a very high level of dexterity and represents a preferential paradigm for dexterous robotic manipulation.

Anthropomorphism is a desirable goal in the design of robotic end-effectors mainly for the following reasons:

- The end-effector can operate in a human-oriented environment (e.g., servicing robots), where tasks may be executed by robots or men as well.
- The end-effector can be tele-operated by a human operator, by means of special-purpose interfaces (e.g., a data-glove), directly reproducing the operator's hand behavior.
- For purposes of entertainment, assistance, and so on, a human-like aspect and behavior may be specifically required, like for humanoid robots.
- For prosthetic devices anthropomorphism is a quite evident design goal. The development of end-effectors for prosthetic purposes [19.13–15] has recently produced so advanced devices that they can be fully considered robotic systems.

While it is difficult to quantify the effective degree of dexterity of a robotic system, its anthropomorphism can be defined in a precise and objective way. In particular, the aspects that mainly contribute to determine the anthropomorphism level of a robotic hand are:

- *Kinematics*: concerning the presence of the main morphological elements (principal upper fingers, secondary upper fingers, opposable thumb, palm).
- *Contact surfaces*: extension and smoothness of the contact surfaces, aspect that reflects on the capability to locate contacts with objects all over the surface of the available links and on the presence of external compliant pads [19.16].

- *Size*: i. e., the size of the robotic hand both referring to the average size of a human hand and the *correct* size ratio between the links.

19.1.2 Dexterity of a Robotic Hand

Besides the *geometrical* reproduction of the human hand, the main research target remains the emulation of those functionalities which make it such a versatile end-effector.

Two are the main capabilities of a human hand:

- *Prehension*, i. e., the hand's ability to grasp and hold objects of different size and shape
- *Apprehension*, or the hand's ability to understand through active touch.

In this sense, the human hand is both an *output* and *input* device [19.17]. As output device, it can apply forces in order to obtain stable grasps or perform manipulation procedures. As input device, it is capable to explore an unknown environment providing information about the state of the interaction with it. The same features are desirable in robot hands. As a matter of fact, the application of robotic systems in unknown environments requires dexterous manipulation abilities to execute complex operations in a flexible way.

A widely accepted definition states that the dexterity of a robotic end-effector is a measure of its capability of changing the configuration of the manipulated object from an initial configuration to a final one, arbitrarily chosen within the device workspace. Generally speaking, with the term *dexterity* we intend the capability of the end-effector, operated by a suitable robotic system, to autonomously perform tasks with a certain level of complexity. An exhaustive review of scientific work developed so far about dexterity of robotic hands, with a quite complete and updated list of references, can be found in [19.18].

Even though the word dexterity itself has a very positive meaning, it may be useful to consider different levels of dexterity, associated with growing complexity and criticality of performable tasks. The dexterity domain for robotic hands can be roughly divided in two main areas, i. e., *grasping* and *internal manipulation*.

Grasping is intended as the capability of constraining objects with a constraint configuration that is substantially invariant with time (the object is fixed with respect to the hand).

Internal manipulation is a controlled motion of the grasped object in the hand workspace, with the constraint configuration changing with time.

Further subdivisions of these two domains have been widely discussed in the literature (different grasp topologies [19.19], different internal manipulation modes based on internal mobility and/or contact sliding or rolling [19.18]).

Although the notion of dexterity is well settled, the way to achieve it remains debated. Factors affecting the actual capabilities of a robotic end-effector are so many that often the analysis and above all the synthesis of dexterous hands do not take into proper consideration some of these elements, namely: morphological features; sensory equipment; control algorithms; task planning strategies; and so on.

19.2 Design of Robot Hands

The mechanical design of an articulated robotic hand can be performed according to many possible design concepts and options, even if a kinematical architecture has already been defined and size and shape specifications imposed. One of the main issues is the design of a proper actuation and transmission system. This aspect is crucial because space and dimensions are usually limited, being in general an anthropomorphic aspect and dimension a design goal to be pursued. Another aspect that is relevant for the design is the adoption of compliant structures (Fig. 19.1), in place of conventional mechanical joints, e.g., rolling pairs [19.20, 21].

Note that, since many solutions and operating concepts can be adopted, what is presented here aims only at illustrating the most significant solutions, and does not pretend to be a complete discussion of all the possible choices.

19.2.1 Actuators Placement and Motion Transmission

In order to actuate the joints of a robot hand, two basic approaches for the placement of the actuators are possible, i. e.:

- Placing the motors as close as possible to each joint, directly in the fingers and sometimes integrating them within the joint itself.
- Placing the motors into the palm or in the forearm; in this case motion is transmitted to each joint by means of (complex) kinematic chains.

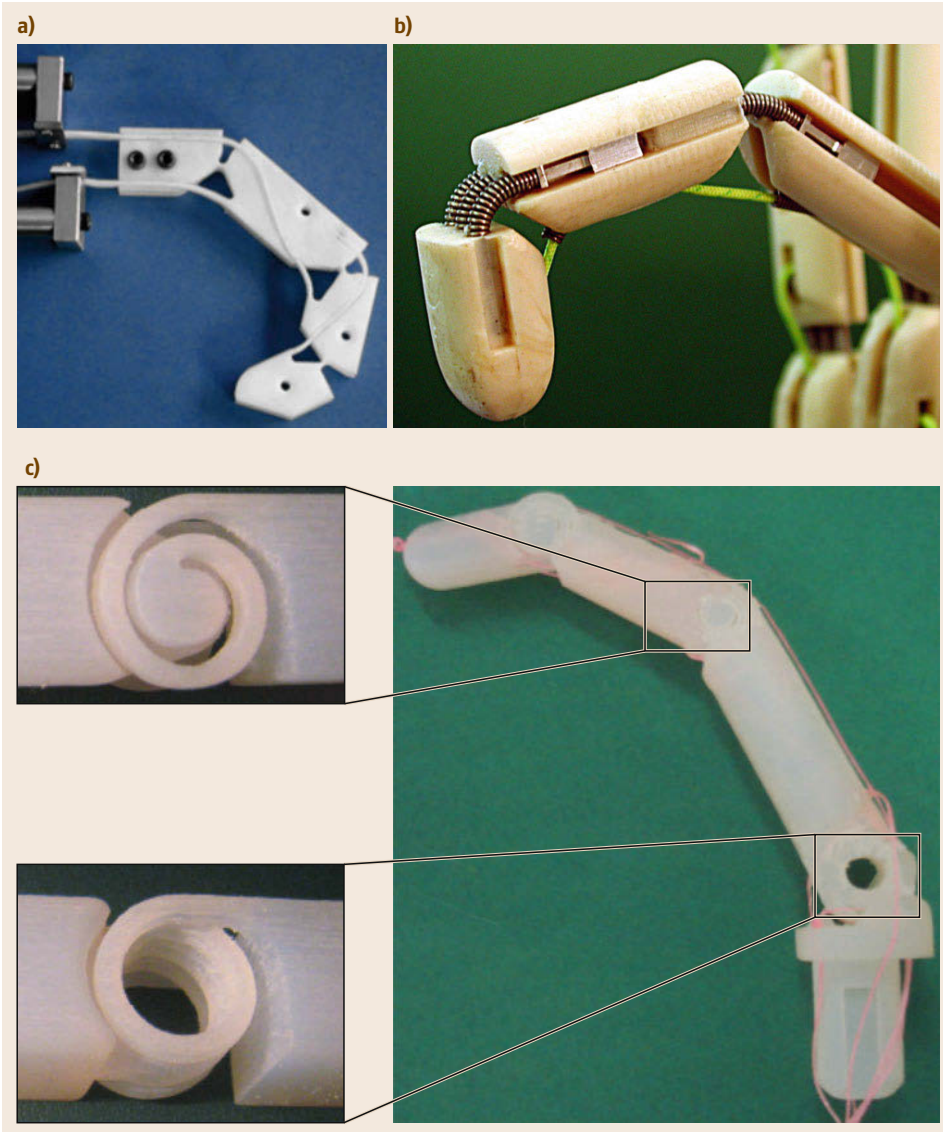


Fig.19.1a–c Three robotic fingers based on compliant joints. **(a)** The finger is obtained in a single teflon piece; **(b)** joint compliance is achieved with metallic springs; **(c)** fast prototyping allows for different compliant mechanisms as joints

In-site actuation can be defined as the case in which the actuator is hosted inside one of the two links connected by the actuated joint or is placed directly inside the joint:

- *Direct-drive actuation*: the actuator is placed directly on the joint, without transmission elements.
- *Link-hosted actuation*: the actuator is placed inside one of the two links constituting the actuated kinematic chain.

In-site actuation simplifies the mechanical configuration of the joint, reducing the transmission chain complexity. In particular, it has the great advantage that the motion of the joint is kinematically independent with respect to other joints. Usually, the size of the finger is imposed by the dimension of the actuators, and for technological reasons it is quite difficult to obtain both an anthropomorphic size and the same grasp strength of the human hand. Furthermore, the motors occupy a large room inside the finger structure, and it is a serious problem to host other elements, like sensors or compliant skin layers. A further negative aspect is that, since the mass of the actuators is concentrated inside the finger, the dynamic behavior of the system and its response bandwidth are reduced.

Nevertheless, the recent advancement of actuator technology enables us to directly implement a quite powerful actuator with reasonable size in each joint.

Avidbots Ex1016
Page 506 of 2154

This built-in actuation has been adopted, e.g., for DLR hand [19.4, 22], ETL hand, Karlsruhe hand, Yasukawa hand, Barrett hand, Gifu hand, U-Tokyo hand, and Hiroshima hand. Since this actuation does not include compliant element like tendons, we can keep a stiff transmission system, which leads to a stable control system even under a high gain (Sect. 19.4). An issue is the routing of wires for both power and signal cables. This issue is more serious in distal joints than for the base joint, since the cables in distal joints produce a relatively large torque disturbance on the first joint, and therefore it is difficult to achieve a precise torque control for this joint.

Remote Actuation

Remote actuation is an alternative solution to in-site actuation. In remote actuation, the joint is driven by actuators placed outside the links connected by the joint itself. Remote actuation requires a motion transmission system, that must pass through the joints between the motor and the actuated joint. In some way, remote actuation must consider the problem of kinematic coupling between the actuated joint and the previous ones. Remote actuation is prevalent in biological structures (e.g., in human hand), where the finger joints are moved by muscles placed on the palm or in the forearm. This human-like approach has been adopted in projects of robotic hands like the UB hand or the National Aeronautics and Space Agency (NASA) Robonaut hand [19.23, 24].

Remote actuation systems can be classified according to the type of adopted transmission elements, i. e., flexible- or rigid-link transmission.

Flexible Link Transmission. Flexible link transmission is based on deformable connections, either flexible or rotational, that can adapt to variations of configuration by changing the transmission path. Linear flexible transmissions are based on flexible elements with translating motion, subject to tension (more frequently) or tension and compression. Two further subcategories can be identified: pulley-routed flexible elements (tendons, chains, belts) or sheath-routed flexible elements (mainly tendon-like elements). Rotational flexible transmissions are based on flexible rotary shafts, that can transmit rotational motion inside the finger structure to the joint, where a final transforming mechanism (a bevel gear or a worm gear) can be used to actuate the joint.

Rigid Link Transmission. Rigid link transmission is mainly based on articulated linkages or on rolling conjugated profiles (mainly gear trains). A further subdivision can be made between parallel and nonparallel axes gear trains, like bevel gears, worm gears, and so on.

19.2.2 Actuation Architectures

Both in-site and remote actuation can be applied according to different types of organization, i. e., by using one ore more actuators for each joint and by making these actuators work in different ways.

In general, we can consider an overall number N of joints for the robotic hand (the wrist joints are not considered) and a number M of actuators that are used to drive, directly or indirectly, the joints. According to different concepts of actuation and transmission, three main categories of actuation schemes can be identified:

- $M < N$: some joints are passive, coupled, or underactuated.
- $M = N$: each joint has its own actuator and there are no passive, coupled or underactuated joints.
- $M > N$: more than one actuator is operating on a single joint.

These architectures strongly depends on the type of motors. In particular, it is possible to recognize two main actuation modalities:

- *Single-acting actuators* – each motor can generate a controlled motion in one direction only: return motion in opposite direction must be obtained by an external action, that can be a passive (e.g., a spring) or an active system (e.g., an antagonistic actuator); this is the case of tendon-based transmission systems.
- *Double-acting actuators* – each motor can generate a controlled motion in both directions and can be used alone to drive the joint or to cooperate with other actuators; in this case the functional redundancy can allow sophisticated drive techniques, like push-pull cooperation.

Each category can be further subdivided. In the following, a brief description of the most frequently adopted schemes is presented.

Single-Acting Actuators with Passive Return Elements

Passive elements, like springs, can store energy during the actuation phase, restituting it during the return stroke (Fig. 19.2a). This mechanism leads to a simplification of actuation scheme, but requires mechanically backdrivable actuators. Other possible drawbacks are related to the loss of available power for the grasp and

Avidbots Ex1016
Page 507 of 2154

the limited response bandwidth in case of low spring stiffness.

Agonistic–Antagonistic Single-Acting Actuators

Two actuators drives the same joint, acting in opposition in different directions (agonistic-antagonistic couple) (Fig. 19.2b). This solution leads to an *N joints-2N actuators* scheme and is quite complex since a large number of actuators must be placed in the hand. On the other hand, it may allow sophisticated control procedures, as both actuators can pull at the same time, with different intensity, generating a driving torque on the joint and a preloading of the joint itself (cocontraction, typical of tendon-driven joints):

- *Pros*: cocontraction strategies, possibility to change the joint stiffness according to the grasping phase and therefore to limit the influence of friction during fast approaching motions; independent position/tension control on each actuator can allow compensation of different path length in case of remote transmission; it is the most flexible solution for driving a joint.
- *Cons*: back-drivability of actuators is required; high difficulty in hosting two actuators for each joint, both in in-site and in remote location; higher control complexity; higher cost.

Single-Acting Actuators Organized According to the Concept of Actuation Net

This is a very interesting case, mimicking biological systems, but has not been implemented yet in robotic hands, except for some preliminary studies. N joints are driven by M actuators, being $N < M < 2N$. Each actuator cooperates in moving more than one joint, thanks to proper net-shaped transmissions:

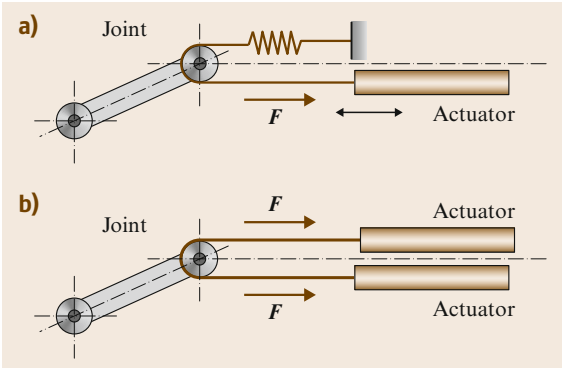


Fig.19.2a,b Single-acting actuator with an antagonist passive element **(a)** and in an agonist-antagonist configuration **(b)**

- *Pros*: cocontraction strategies, possibility to change the joint stiffness according to the grasping phase and therefore to limit the influence of friction during fast approaching motions; reduced number of actuators with respect to the $2N$ actuators scheme.
- *Cons*: back-drivability of actuators is required; high complexity of the kinematic scheme and therefore high complexity in control.

The simplest case of actuation net is represented by the so called $N + 1$ *actuation* (being N in this case the number of joints of a finger), frequently adopted in practice (Fig. 19.3). In this case, all actuators are coupled, and therefore a damage of any of them will result in a general failure.

Double Acting Actuators with $M < N$

In this case, the number of actuators is less than the number of joints. With reference to a single motor and several joints, two main subcases can be defined:

1. The joints are kinematically coupled, in a fixed or variable way, so that the number of degrees of freedom of this subsystems is reduced to one.
2. The joints are selectively actuated by the motor, according to an active or passive selection subsystem.

The former case can be further subdivided:

- *Joints kinematically coupled in a fixed way*: In this kind of kinematical configuration, each motor can move more joints connected by rigid mechanisms with fixed transmission ratios. A typical application is obtained with the use of a gear train: the first link is directly actuated by a motor, while a gear transmission between a wheel fixed to the frame and a final wheel connected to the joint generates the relative motion of the second link (Fig. 19.4a). Should the motion of two parallel fingers be required, their connection could be easily obtained mounting two gear wheels on the same shaft. Another very common way to obtain this kind of kinematical linkage is to use tendon driven devices

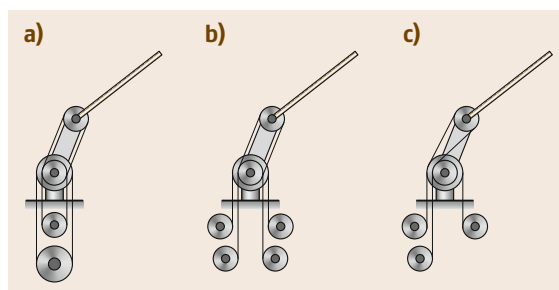


Fig.19.3a–c Remote actuation. **(a)** N-type, **(b)** $2N$-type, **(c)** $N + 1$-type

Avidbots Ex1016
Page 508 of 2154

as shown in Fig. 19.4b. In artificial hand design, the main advantage when using joints driven by fixed-ratio mechanisms is the possibility to know and control the position of the second link. A disadvantage is that this kind of mechanisms does not adapt to the shapes of the grasped objects, and this may cause grasp instability.

- *Joints coupled in a non-fixed way*: This is the case of underactuated mechanisms and deformable passive-driven joints. A mechanism is said to be *underactuated* when the number of actuators is smaller than the number of degrees of freedom. When applied to mechanical fingers, this concept may lead to shape adaptation, i. e., underactuated fingers can envelope the objects to be grasped and adapt to their shape even with a reduced number of actuators. In order to obtain a statically determined system, elastic elements and mechanical limits must be introduced in underactuated systems (simple linear spring are often used). In the case of a finger closing on an object, for instance, the configuration of the finger is determined by the external constraints associated with the object. An example of an underactuated two-degrees of freedom finger is shown in Fig. 19.5 [19.25]. The finger is actuated through the lower link, and a spring is used to maintain the finger fully extended. A mechanical limit is used to keep the phalanges aligned under the action of this spring when no external forces are applied on the phalanges. Since the joints cannot be controlled independently, the behavior of the finger is determined by the design parameters (i. e., the geometric and the stiffness properties). Hence, the choice of these design parameters is a crucial issue.
  Another approach consists in coupling the motion of two adjacent joints by means of deformable linkages. This feature introduces in the kinematical chain the needed compliance to fit to the shapes of the grasped objects. A very simple mechanism of this category is reported in Fig. 19.6. Structurally it is similar to the mechanisms based on a fixed coupling, the only important difference is the addition of a spring to give extensibility to the tendon. This spring allows to decouple the motion between the first and second link when an external force is applied to the distal one. This solution is widely used: a well known example is the DLR hand. The benefits of this solution are mainly due to the possibility to fit to the shapes of objects. A design problem is the choice of the stiffness of the deformable element in order to achieve at the same time a strong grasp and a good shape adaptability.
- *Joints selectively driven by only one motor*: With this solution, the motion generated by only one (large) motor is transmitted and distributed to several joints. Actuation and control of each joint is obtained by means of insertion-disinsertion devices like self-acting or commanded clutches.

Double-Acting Actuation, with $M = N$

This is a very common case: each joint is driven in both directions by the same actuator. The achievable performances are therefore similar (equal) in both directions, but particular attention must be paid to backlash, and it is usually necessary to preload the transmission system. In particular, preload is mandatory in case of transmission by means of flexible elements like tendons (Fig. 19.3a). Furthermore, the adoption of a closed-

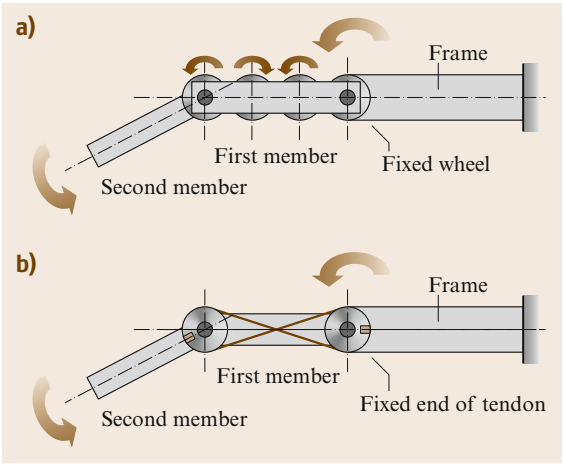


Fig. 19.4 Double-acting actuator with $N = M$ based on gears **(a)** and tendons **(b)**

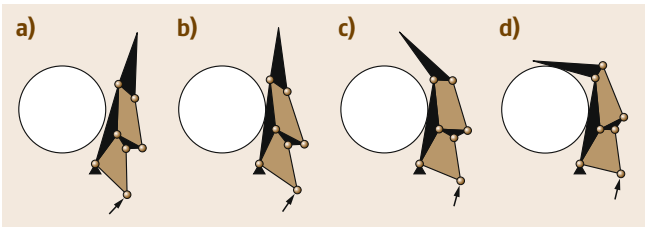


Fig.19.5a–d Grasping sequence performed by a finger based on underactuated mechanism

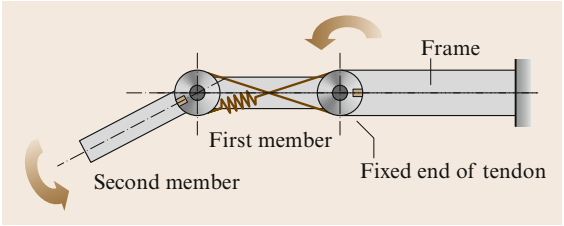


Fig. 19.6 Joints coupled in non fixed way

loop tendon transmission requires that the overall length of the tendon route must be kept constant, according to the concept that winded and unwinded parts of the tendon on the motor pulley have the same length; this involves the need of length-compensating mechanisms (e.g., pulley-trains, cams) every time that changes in the geometry of the finger cause a differential displacement of the tendons. In spite of this required complexity, this actuation scheme has been widely used, with simple pulley routing (UB hand, Okada hand, ...), or sheath-routing (Salisbury hand, Dipartmento di Informatica Sistemica e Telematica (DIST) hand), that has a simpler mechanical structure but must face the problem of sheath-tendon friction (application of high preload is not convenient in this case).

19.3 Technologies for Actuation and Sensing

In this Section, a brief description of the main issues related to technological aspects of actuation and sensing for robot hands is reported. A more general presentation and detailed description of these aspects is given in Part A (Chaps. 4 and 5) and Part C (Chap. 28).

19.3.1 Actuation

Electrical actuators are without doubt the most common choice for actuating robot hands. As a matter of fact, electric motors have very good performance in terms of position/velocity control, have a reasonable mass/power ratio, and are a very common technology, that does not require external devices (as for hydraulic or pneumatic actuators). However, there are several other possibilities. For example: ultrasonic motors (Keio hand [19.26]), chemical actuators, pneumatic actuators (McKibben in the Shadow hand [19.27]; VIDEO 753), spring based actuators (as for the 100G Capturing Robot [19.28] VIDEO 755), twisted string actuators (Fig. 19.7) [19.29] (Dexmart hand [19.30]; VIDEO 767), and others.

In particular, for pursuing quick responses, either pneumatic or spring-based actuators may be good solutions, although it should be noted that a braking system with quick response is essential for achieving good position controllability for this type of actuators.

19.3.2 Sensors

In robot hands, as in other robotic devices, sensors can be classified in two main categories: *proprioceptive* and *exteroceptive* sensors. The first type of sensors measures physical information related to the state of the device itself (e.g., position, velocity, and so on), while the second one is devoted to the measurement of data related to the interaction with objects/environment (e.g., applied forces/torques, friction, shape, and so on).

Joint Position/Velocity Sensors

For control purposes, there is the obvious necessity of measuring position/velocity of the actuated joints. A major problem consists in the limitation of the available space, both for the sensors and for the wires. Different technological solutions can be adopted, but a rather common choice is based on Hall-effect sensors, that are sufficiently small, precise and reliable for this type of application. In case of remote actuation, there is the possibility of having two position/velocity sensors for each joint: one located in the actuator (e.g.,

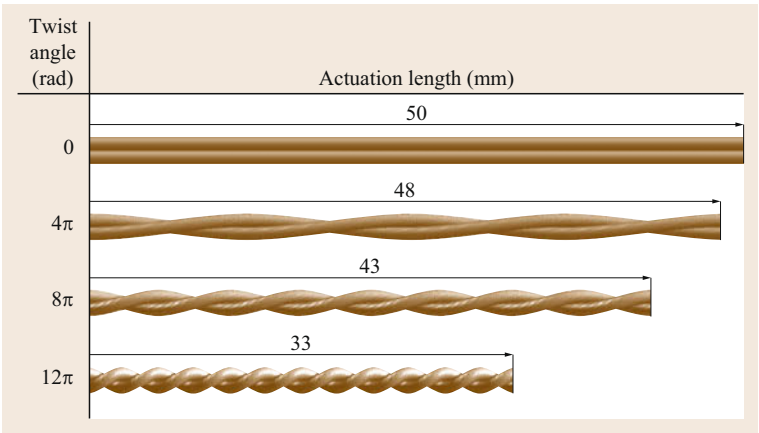


Fig. 19.7 The twisted string concept: by twisting the string, its twisted string concept length is reduced transforming a rotational motion into a linear one

Avidbots Ex1016
Page 510 of 2154

an encoder) and one placed in the joint itself, often necessary because of the non linearities introduced by the transmission system (elasticity, friction, and so on). Quite often, this latter sensor is specifically designed and implemented for the given hand, being commercially available sensors too large and not suitable for installation in the joints.

Tendon Tension Sensor and Joint Torque Sensor

It is well known that humans can control finger tip compliance as well as finger tip force by controlling voluntary muscles. In remote actuation, it is essential to measure the tendon tension for two main reasons: for compensating the friction existing in the transmission system, and for measuring the external contact force. Figure 19.8 shows a way for measuring the tendon tension where the tendon is pressed by an elastic plate with a strain gauge. When a tension is applied to the tendon, the sensor measures a force composed of axial and bending force components. The displacement of the elastic plate due to the axial force component is negligibly small compared with that due to the bending force component. As a result, the bending force component generates a bending deformation for the elastic plate. This deformation is transformed in an electric signal by means of proper transducers, such as strain gauges attached on the surface of the plate or optoelectronic components [19.31, 32]. Now, suppose N-type actuation with two tension sensors, as shown in Fig. 19.9, where joint torque τ is given. Note that $\tau = r(T_1 - T_2)$ where r, T_1, and T_2 are the pulley radius and tendon tensions, respectively. Since we can measure e_1 and e_2 corresponding to T_1 and T_2, τ can be obtained by feeding both e_1 and e_2 into the differential circuit. This approach, however, includes a couple of issues. The main problem is the plastic deformation of the sensor plate under an extreme large pretension. Once such a plastic deformation has happened, the sensor will never work appropriately anymore. Another minor issue is are that two sensors are always necessary for measuring a joint torque. To cope with these issues, the tension-differential-type torque sensor [19.28] can be used as shown in Fig. 19.10. The sensor is designed with just a single body and it partially includes an elastic part where at least one strain gauge is attached. The working principle of the sensor, shown in Fig. 19.10a, supposes that a torque is applied to the joint. This means that T_1 and T_2 have different values. This difference causes a bending force around the strain gauge. The key is that the bending force is kept to zero even under an extremely large tension as far as no joint torque is given. Therefore, we are completely released from the plastic deformation of the elastic plate due to pretension. Furthermore, the sensor is constructed with just a single body. There are couple of variations in this type of torque sensor. As decreasing the pulley distance in Fig. 19.10a, the sensor eventually results in the single-pulley-version with zero distance, as shown in Fig. 19.10b. The single-pulley-version has been implemented into Darmstadt

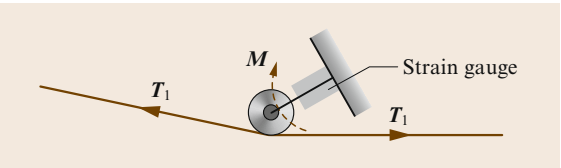


Fig. 19.8 Tendon tension sensing

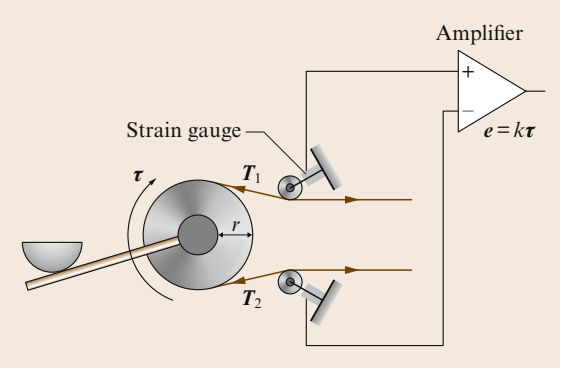


Fig. 19.9 Tension sensor based torque sensing

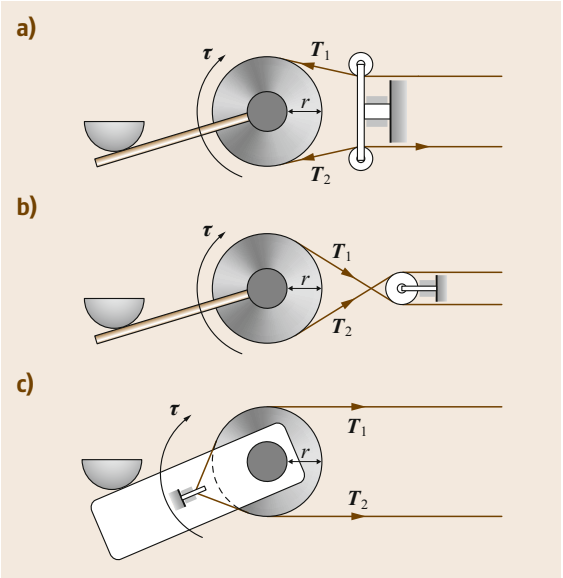


Fig.19.10a–c Tension-differential type (TDT) sensor. **(a)** Double pulley version, **(b)** single pulley version, **(c)** pulley-less version (after [19.28])

Avidbots Ex1016
Page 511 of 2154

hand [19.33] and MEL hand [19.34]. Furthermore, if the sensor is built in the finger link connected by the concerned tendon, there is no relative motion between the sensor and the tendon. As a result, we can remove the pulley, as shown in Fig. 19.10c. This is called as the pulley-less version and has been implemented into Hiroshima hand. The tension-differential-type torque sensor will be a powerful tool for measuring a tendon drive joint.

Finger Tip Tactile (or Force) Sensors

Most robot manipulation and assembly tasks would benefits of the utilization of tactile sensory information. When lifting an object, tactile sensing could detect the onset of slip in time for corrective action to be taken. In addition to the contact point between the finger tip and the object, several objects properties, such as friction coefficient of the object surfaces, surface texture, and weight can be determined by utilizing a finger tip tactile (or force) sensor. A six-axis force sensor allows us to detect contact point as well as contact force between finger and environment, if a single contact is assumed. For the finger model as shown in Fig. 19.11, the following relationship between the sensor output and contact force may be defined

$$\boldsymbol{F}_\mathrm{s} = \boldsymbol{f}\,, \tag{19.1}$$

$$\boldsymbol{M}_\mathrm{s} = \boldsymbol{x}_\mathrm{c} \times \boldsymbol{f}\,, \tag{19.2}$$

where $\boldsymbol{f} \in \mathbb{R}^3$, $\boldsymbol{F}_\mathrm{s} \in \mathbb{R}^3$, $\boldsymbol{M}_\mathrm{s} \in \mathbb{R}^3$, and $\boldsymbol{x}_\mathrm{c} \in \mathbb{R}^3$ are the external force vector, the force vector measured by the six-axis force sensor, the moment vector measured by the six-axis force sensor, and the position vector indicating the contact position, respectively. From the first equation, we can directly obtain the contact force. Putting $\boldsymbol{F}_\mathrm{s}$ into the second equation leads to $\boldsymbol{M}_\mathrm{s} = \boldsymbol{x}_\mathrm{c} \times \boldsymbol{F}_\mathrm{s}$. $\boldsymbol{x}_\mathrm{c}$ is determined in such a way that $\boldsymbol{M}_\mathrm{s} = \boldsymbol{x}_\mathrm{c} \times \boldsymbol{F}_\mathrm{s}$ may be satisfied. For a finger with convex object, we have always two mathematical solutions as shown in Fig. 19.12a where the meaningful solution is the one satisfying $\boldsymbol{f}^\mathrm{t}\boldsymbol{n} < 0$, n being the outward normal direction to the finger's surface (a finger can only push the object). However, for a finger with concave shape, we have at least four mathematical solutions, as shown in Fig. 19.12b where two of those are physically possible. A finger with the six-axis force sensor located in the fingertip, Fig.19.12.c, can avoid multiple solutions. On the other hand, only forces applied to the fingertip can be detected, and if more links are in contact with the object it would be necessary to have a force/torque sensor placed in each of them.

This type of solution, i. e., a multiaxis sensor for measuring not only forces and torques but also the position of the contact point, is known in the literature as the *intrinsic tactile* (IT) principle [19.35]. In general, with respect to the use of traditional tactile sensors, see later, it leads to a simplification in the design since it requires less wires and connections for the sensor.

Tactile Sensors

Another important class of sensing devices consists of tactile sensors, which are used for several purposes, such as shape recognition, contact point determination, pressure/force measurement. A number of tactile sensors have been proposed in the literature, with several different solutions concerning the implementation features: optical, piezoresistive, piezoelectric, and so on. References [19.36, 37] give an overview on technologies and applications.

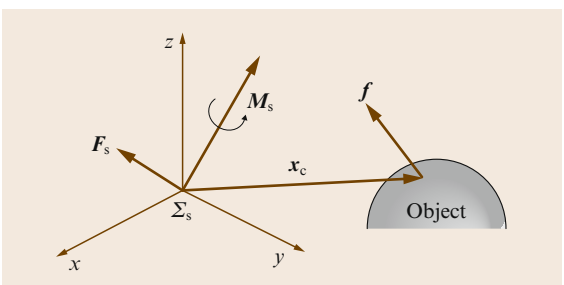


Fig. 19.11 Sensor coordinate system $\sum_\mathrm{S}$

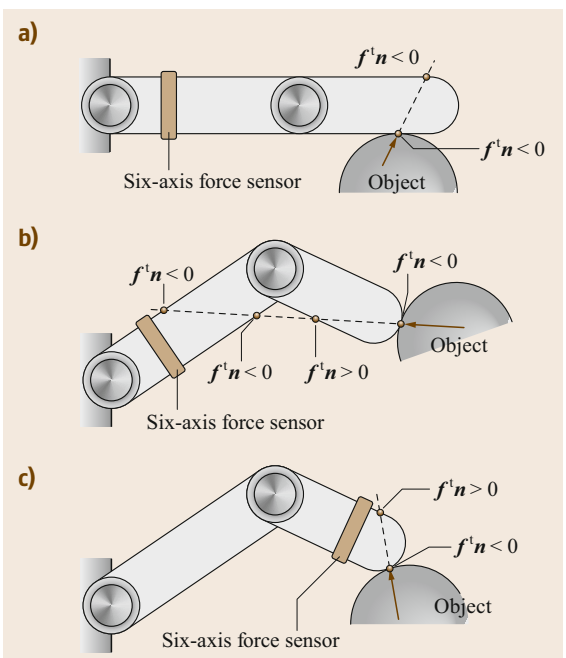


Fig. 19.12 Interpretation of solutions. **(a)** Convex shape, **(b)** finger with a concave shape, **(c)** sensor located in the fingertip

Avidbots Ex1016
Page 512 of 2154

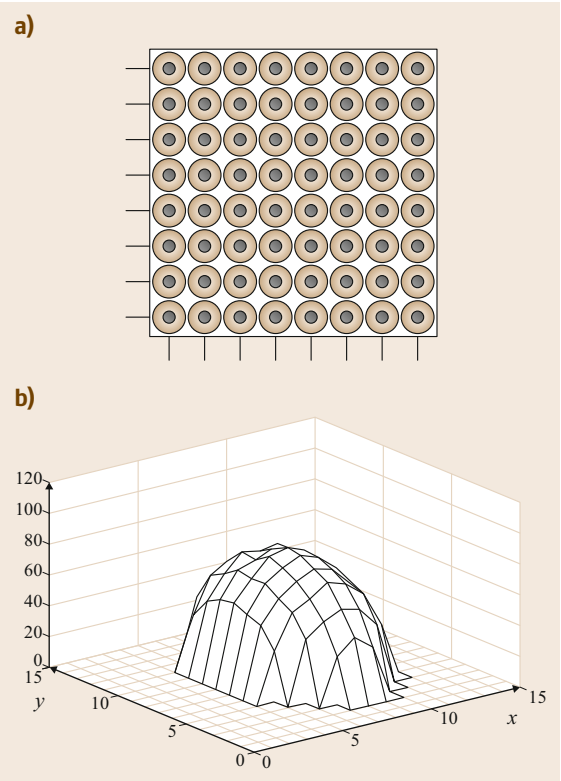


Fig.19.13a,b A Tactile sensor. **(a)** Scheme of a tactile sensor, **(b)** example of data from a tactile sensor

Tactile sensors have been introduced in robotics since the late 1970s. Nowadays, as the force sensors, also tactile sensors are commercially available devices. Probably, they represent the most commonly adopted sensorial class for industrial grippers, even though they are often used as advanced *on/off* devices to check whether a grasp or contact condition occurs.

Usually, they consist in a matrix (array) of sensing elements. Each sensing element is usually referred to as a *taxel* (from *tactile element*), and the whole set of information is called a *tactile image*, Fig. 19.13. Main goal of this class of sensors is to measure the map of pressures over the sensing area.

In general, the types of information that may be obtained from a tactile sensor are:

- *Contact*: This is the most simple information given by the sensor, concerning the presence or absence of a contact.
- *Force*: Each sensing element provides an information related to the amount of locally applied force, which can be used in several manners for successive elaborations.
- *Simple geometrical information*, i. e., position of the contact area, geometrical shape of the contact itself (planar, circular, and so on).
- *Main geometrical features of the object*: By proper elaborations of the data of the taxels, it is possible to deduce the type of object in contact with the sensor, for example a sphere, a cylinder and so on (data relative to the 3-D (three-dimensional) shape).
- *Mechanical properties*, such as friction coefficient, roughness, and so on. Also thermal properties of the object may be measured by a tactile sensor.
- *Slip condition*, i. e., the relative movement between the object and the sensor.

Several technologies have been adopted for the design of tactile sensors, ranging from piezoresistive to magnetic, to optical effects, and so on. Among the most common, one can mention:

- Resistive and conductive effect
- Electromagnetic effect
- Capacitive effect
- Piezoelectric effect
- Optical effect
- Mechanical methods.

Each of these technologies has positive and negative aspects. Common drawbacks, however, are the size of these sensors, usually quite large in comparison with the available space, and the necessity of a high number of electrical connections.

19.4 Modeling and Control of a Robot Hand

The dynamic model of a robot hand with in-site actuation is very similar to the model of a traditional (industrial) robot, and the hand can be considered as a collection of robot manipulators. On the other hand, remote actuation introduces some peculiar features that have to be carefully considered. In particular, the problems tied to nonlinear phenomena (e.g., friction and backlash), compliance of the transmission system, and noncolocation of sensors and actuators are very critical for the design of the control. Moreover, the use of single-acting actuators, such as tendon based actuation systems, requires the adoption of proper control techniques, which allow the imposition of the desired torque at each joint of the hand, despite the coupling among them.

Avidbots Ex1016
Page 513 of 2154

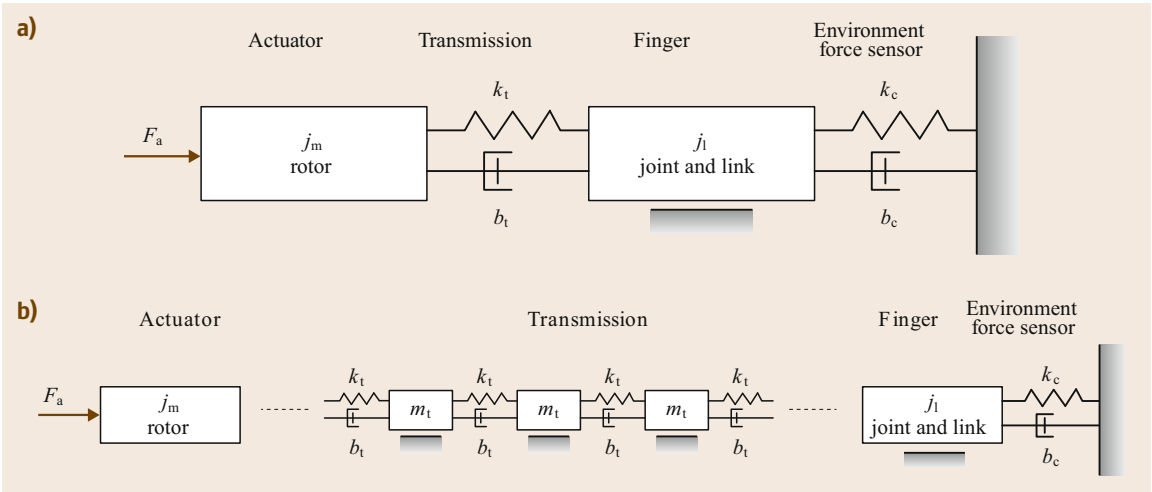


Fig.19.14a,b Model of a robot joint (**a**) with transmission flexibility, and (**b**) with tendon based transmission

19.4.1 Dynamic Effects of Flexible Transmission Systems

The transmission system of robot hands with remote actuation is usually characterized by an high level of friction and non negligible dynamic effects which complicate the control problem. A simple representation considers a single axis motion with two inertial elements linked by an elastic transmission. This is the typical representation of elastic joints in which the former element represents the motor inertia, while the latter is related to the inertial properties of the actuated joint/link (Fig. 19.14a). More complex models assume a dynamic model for the transmission system, i. e., the classical representation of tendons based on the serial repetition of masses linked by springs/dampers, reported in Fig. 19.14b. These simple models are particularly useful to understand some drawbacks and limitations due to the fact that actuation system and actuated element are located in two different places and the motion is transmitted by a nonideal (that is not purely static) element. If we consider the capability of the fingers' joint of applying a force on the environment, the effect of the transmission system on the open loop response of the system modeled as in Fig. 19.14a, are a noticeable reduction of the bandwidth, and an important phase delay between the input F_a (the force applied by the motor) and the output F_c (the force exchanged at the contact). As shown in Fig. 19.15, the open-loop transfer function

$$\frac{F_a}{F_c} = \frac{(b_c s + k_c)(b_t s + k_t)}{\left[j_l s^2 + (b_t + b_c)s + k_t + k_c\right](j_m s^2 + b_t s + k_t) - (b_t s + k_t)^2} \tag{19.3}$$

is characterized by four poles that, for growing values of the transmission stiffness k_s, move from their initial locations (that depend on the values of physical parameters j_l, j_m, etc., although for $k_s = 0$ at least one pole is in the origin of the Gauss plane) towards the poles of a system with an infinitely rigid transmission (for k_c tending to ∞, two poles go to infinity) (and a total inertia given by the contributions of both the motor and the link) whose transfer function is

$$\frac{F_a}{F_c} = \frac{(b_c s + k_c)}{(j_l + j_m)s^2 + b_c s + k_c} \tag{19.4}$$

As a consequence, the bandwidth of the system with flexible transmission, that for high values of k_s approx-

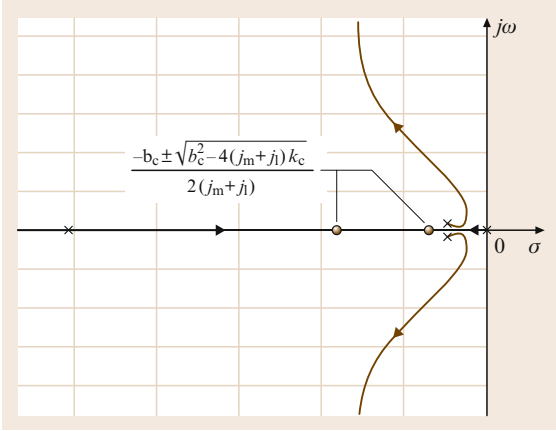


Fig. 19.15 Root contour of the transfer function (19.3) with variable k_t

Avidbots Ex1016
Page 514 of 2154

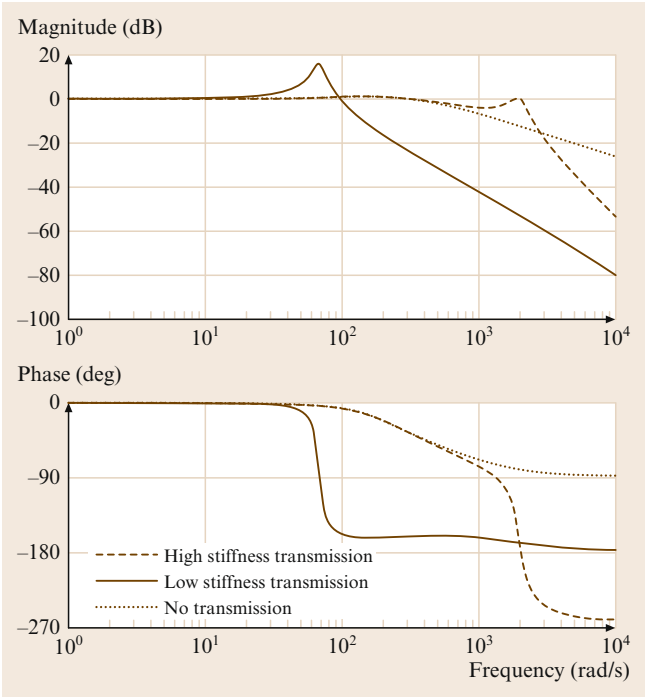


Fig. 19.16 Bode plots of the open loop transfer function (19.3) for low stiffness values (*continuous line*) high stiffness value (*dashed line*) and with no transmission (*dotted line*) ►

imates those of (19.4), decreases when the compliance of the transmission is not negligible, see the Bode plots reported in Fig. 19.16. The bandwidth of the system with flexible transmission is strongly affected by the location of the speed reducer: when the reducer is placed at the joint the bandwidth is K_r times (K_r is the reduction ratio) higher than the one achievable with the reduction applied directly on the motor [19.38] (Fig. 19.17). Moreover, it is worth to notice that for low level of the stiffness k_t a sharp phase drop occurs at the frequency of the flexible mode. Therefore, there are some frequencies (relatively low) at which the force applied by the motor and the one measured with a sensor in the finger's joint are completely out of phase. These effects, which may cause the instability of the overall system under force control (or impedance control) are referred to as *noncolocation*. In general, when actuators and sensors are physically located at different points of a flexible structure (or a structure with flexible transmission), there will be unstable modes in the closed-loop system [19.39].

From the control viewpoint, the problem of mechanical transmission flexibility is further exasperated by the non linear frictional phenomena that inevitably affect remote actuation and motion transmission. As a matter of fact, the linear viscous friction, represented in Fig. 19.14 by the damping coefficient b_t, is accompanied by stiction and Coulomb friction, both of which are discontinuous at zero velocity (Fig. 19.18). These nonlinearities may cause limit cycles and input-dependent stability, and must be accurately taken into account in the design of the robot hand structure as well as of its control architecture [19.40]. For instance, in the design of the Utah/MIT dexterous hand, depicted in Fig. 19.19, in order to reduce static friction, the idea of using tendon sheaths was abandoned in favor of pulleys [19.3]. In order to find an optimal trade-off between complexity and reliability of the mechanical arrangement and achieved friction level, a number of solutions, which combine sheaths and pulleys for routing the tendon from actuators to fingers' joints has been adopted in the design of robot hands, e.g., the Stanford/JPL hand (VIDEO 751) and the UB hand 3 reported in Fig. 19.20. This device is characterized by an extremely simple structure, with the tendons completely routed within sheaths, but on the other hand the friction cannot be absolutely neglected and a precise modeling of the interaction between the tendons and the tube is necessary for control purposes [19.41].

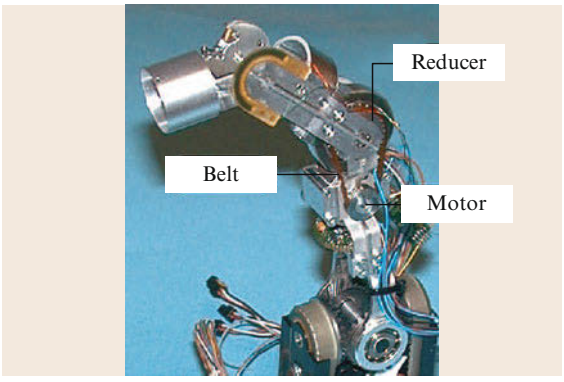


Fig. 19.17 Reducer location on the medial joint of the DLR hand II

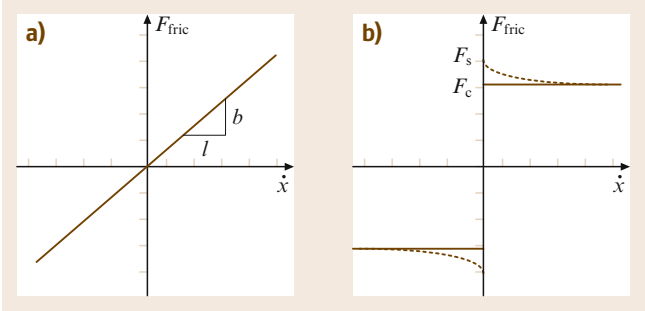


Fig.19.18a,b Frictional phenomena: viscous friction **(a)** and stiction and Coulomb friction **(b)**

Avidbots Ex1016
Page 515 of 2154

19.4.2 Transmission Model of Tendon-Outer-Tube System

This system can be modeled as shown in Fig. 19.21 where T_{in}, T_{out}, T_0, ξ_{in}, ξ_{out}, R_i, x, and L are the tension at the input side, the tension at the output side, the initial pretension, the displacement at the input side, the displacement at the output side, the local radius of routing, the coordinate system along the wire, and the length of the tendon, respectively. The relationship between the tension at output and the input displacement is given by [19.42],

$$T_{out} - T_0 = K_t(\xi_{in}\phi_B)\,, \tag{19.5}$$

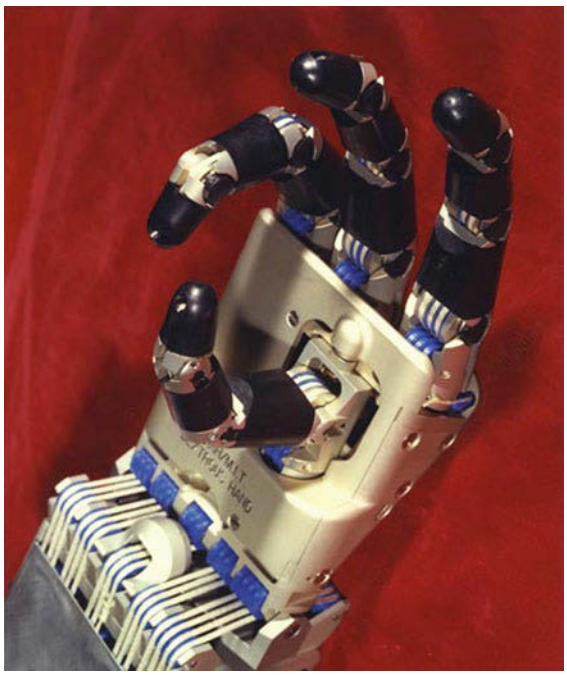


Fig. 19.19 The Utah/MIT robotic hand

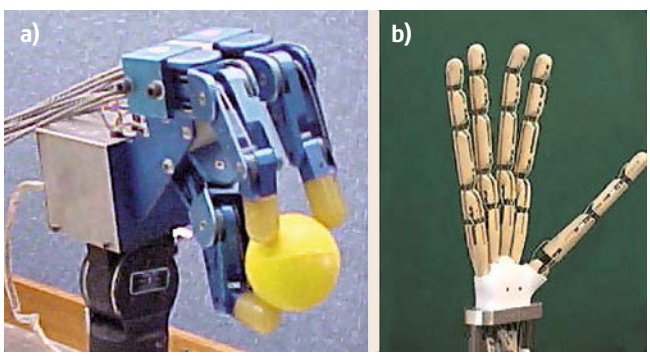


Fig.19.20a,b Tendon based robot hands: Stanford/JPL hand (**a**) and UB hand 3 (**b**)

where K_t and ϕ_B are the total stiffness and the equivalent backlash, respectively, and those are given by,

$$\frac{1}{K_t} = \frac{1}{K_e} + \frac{1}{K_s} + \frac{1}{K_{ap}} \tag{19.6}$$

$$K_{ap} = K_w \frac{\lambda}{\exp(\lambda) - 1} \tag{19.7}$$

$$\phi_B = \frac{T_0 L}{EA} \cdot \frac{\exp(\lambda) - \lambda - 1}{\lambda} \tag{19.8}$$

$$\lambda = \sum |\beta_i| \mu \operatorname{sgn} \xi_{in} \tag{19.9}$$

where K_e, K_s, K_w, K_{ap}, μ, E, A and β_i are the stiffness of environment, the stiffness of force sensor, the apparent stiffness of the tendon, the friction coefficient, Young's modulus, the cross sectional area, and the bending angle of each segment of tendon, respectively. For example,

$$\sum |\beta_i| = 2\pi$$

for the case given in Fig. 19.21. As can be seen from this example, the friction related parameter λ increases dramatically when the tube is heavily bent. While we have a big advantage of choosing a free route for the power transmission, it brings a large nonlinearity for the transmission system. We would note that while both the apparent stiffness of the tendon and the equivalent backlash vary depending upon λ which is the function of the curvature of the route as well as the friction coefficient, ϕ_B and K_{ap} result in $\phi_B = 0$ and $K_{ap} = K_w$ under $\mu = 0$. From the view point of control, such hysteresis is, of course, not desirable. To cope with these issues, each tendon should be designed as short as possible, so that we may keep high stiffness and small backlash in the transmission system.

19.4.3 The Control Through Single-Acting Actuators

The use of single-acting actuators (i. e., standard motors with tendinous transmission), which are commonly

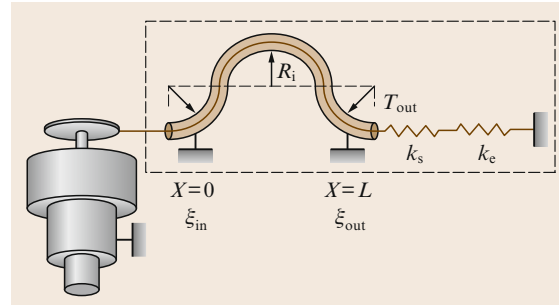


Fig. 19.21 Model of tendon-outer-tube transmission

Avidbots Ex1016
Page 516 of 2154

assumed in the design of robot hands with remote actuation, requires the adoption of specific control techniques in order to guarantee the desired torques at the joints and to maintain at each time a positive tension on the tendons. To this purpose, the tendons are treated like inelastic frictionless elements and the problem is coped in a way completely decoupled from the issue of the device stability, discussed in previous sections.

A tendon, routed in the finger structure through sheaths or/and pulleys, can be modeled by means an *extension function* $l_i(\theta)$ [19.43] relating the joints' configuration with the tendon elongation. In the case of the tendon network represented in Fig. 19.3, the extension functions of the three tendons have the form

$$l_i(\boldsymbol{\theta}) = l_{0i} \pm R\boldsymbol{\theta}_1 \pm R\boldsymbol{\theta}_2 \,,$$

where R is the radius of the pulleys and

$$\boldsymbol{\theta} = [\boldsymbol{\theta}_1 \boldsymbol{\theta}_2]^{\mathrm{T}}$$

is the vector of the joint variables. Once the extension function has been determined, it is straightforward to derive the relationship between tendons forces and resulting joints torques. As a matter of fact, the relation between the joint speeds $\dot{\boldsymbol{\theta}}$ and tendon speeds $\dot{\boldsymbol{l}}$ can be deduced by simply differentiating the expression of extension functions

$$\dot{\boldsymbol{l}} = \frac{\partial \boldsymbol{l}}{\partial \boldsymbol{\theta}}(\boldsymbol{\theta})\dot{\boldsymbol{\theta}} = \mathbf{P}(\boldsymbol{\theta})\dot{\boldsymbol{\theta}} \,. \tag{19.10}$$

Because of the conservation of the power, from (19.10), one can achieve

$$\boldsymbol{\tau} = \mathbf{P}^{\mathrm{T}}(\boldsymbol{\theta})\boldsymbol{f} \,, \tag{19.11}$$

where $\boldsymbol{\tau}$ are the torques exerted on the joints, and $\boldsymbol{f}$ are the force applied by tendons. From (19.11) it results that the force transmitted by a tendon may affect (and, in general, will affect) more than one joint.

In order to guarantee the possibility of exerting joint torques in every direction under the constraint of pure tensile forces, for any $\boldsymbol{\tau} \in \mathbb{R}^n$ it must exist a set of forces $\boldsymbol{f}_i \in \mathbb{R}^m$ (n and m are respectively the number of joints and the number of tendons) such that

$$\boldsymbol{\tau} = \mathbf{P}^{\mathrm{T}}(\boldsymbol{\theta})\boldsymbol{f} \quad \text{and} \quad \boldsymbol{f}_i > 0, i = 1, \ldots, m \,. \tag{19.12}$$

In this case the tendon network is said *force closure*. If the condition expressed by (19.12) is verified, given a desired torque vector $\boldsymbol{\tau}$ it is possible to compute the force that the actuators must provide to the tendons according to

$$\boldsymbol{f} = \mathbf{P}^{\dagger}(\boldsymbol{\theta})\boldsymbol{\tau} + \boldsymbol{f}_{\mathrm{N}} \,, \tag{19.13}$$

where

$$\mathbf{P}^{\dagger} = \mathbf{P}(\mathbf{P}^{\mathrm{T}}\mathbf{P})^{-1}$$

is the pseudo-inverse of the coupling matrix $\mathbf{P}^{\mathrm{T}}$ and

$$\boldsymbol{f}_{\mathrm{N}} \in \mathcal{N}(\mathbf{P}^{\mathrm{T}})$$

is a vector of *internal forces* that insures that all tendon tensions are positive. In general, internal forces will be chosen as small as possible, so that the tendons are always taut but are not subject to excessive strains.

19.4.4 Control of a Robot Hand

The modeling and control aspects described in the previous Sections, although very important and fundamental, can be considered as a sort of *low level* problems in the control of a robot hand, in the sense that they are related to the specific physical properties of the device.

There are also other problems that must be faced and solved in order to operate in a profitable manner with a multifingered hand. These problems are solved by a proper design of a *high level* control for the hand, that must take into account the interaction of the hand with the objects and more in general with the environment. In this context, general aspects that must be considered are: the control of forces/torques applied at the contact points, the necessity to model contact compliance/friction effects, the type of mobility both for the fingers and at the contact (rolling, sliding, ...), a suitable planning algorithm for grasping and/or manipulating the objects, and so on.

These problems are illustrated in detail in Part C, Chaps. 37–39.

19.5 Applications and Trends

In the industrial environment, simplicity and cost are the main guidelines for the design of end-effectors, and therefore simple devices, as open-close grippers, are very common and widely used. This situation has led during the years to the development of a number of special-purpose devices, optimized for single

Avidbots Ex1016
Page 517 of 2154

specialized operations but not suitable for other tasks. At the moment, dexterous multifingered hands have not really been applied to any major application, mainly because of problems of reliability, complexity, cost.

On the other hand, more and more operations are currently envisaged for robots working in environments designed for, and utilized by, human operators. Entertainment, maintenance, space/underwater applications, help to disable persons are just a few examples of use of robotic systems in which interaction with tools and objects designed for human beings (or directly with them) is implied. In all these circumstances, the robot must be able to grasp and manipulate objects (different in dimension, shape, weight, ...) similarly to humans, and therefore a robot hand, with a proper number of fingers and joints and also with an anthropomorphic appearance, seems to be the most adequate solution.

There are several projects aiming at developing anthropomorphic robots. Among others, one can mention the NASA/JPL Robonaut [19.24], Fig. 19.22, the devices developed at the DLR, the several projects on humanoid robots currently under development.

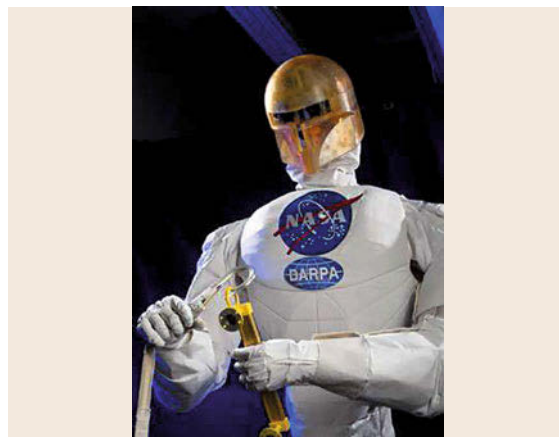


Fig. 19.22 The NASA/JPL Robonaut

19.6 Conclusions and Further Reading

The design of multifingered robot hands has attracted the interest of the research community since the early days of robotics, not only as a challenging technical problem itself but, probably, also because of anthropomorphic motivations and the intrinsic interest for a better knowledge of the human beings. In the last decades, has previously discussed, several important projects have been launched, and important examples of robot hands developed. Nevertheless, the current situation is that reliable, flexible, dexterous hands are still not available for real applications. For these motivations, it is easy to foresee also for the future a consistent research activity in this fascinating field, with developments at the technological (sensor, actuator, material, ...) and methodological (control, planning, ...) level. Important connections with other scientific fields are also expected, as for example with cognitive science.

Being this research area so wide, it is not simple to suggest to interest readers further readings, except for quite classical books such as [19.43–45]. As a matter of fact, depending on the specific research area, many publications are available, although often not organized as reference books, but mainly as technical papers published in journals or presented at international conferences. Moreover, since hundreds of new papers are published every year covering the different aspects of this robotic field, it is really quite difficult, and also not fair, to give at the moment specific suggestions for further readings. We can only refer to the citations already provided in the references.

Video-References

VIDEO 749 The PISA-IIT SoftHand
available from http://handbookofrobotics.org/view-chapter/19/videodetails/749

VIDEO 750 The PISA-IIT SoftHand
available from http://handbookofrobotics.org/view-chapter/19/videodetails/750

VIDEO 751 The Salisbury Hand
available from http://handbookofrobotics.org/view-chapter/19/videodetails/751

VIDEO 752 The Barrett Hand
available from http://handbookofrobotics.org/view-chapter/19/videodetails/752

VIDEO 753 The Shadow Hand
available from http://handbookofrobotics.org/view-chapter/19/videodetails/753

Avidbots Ex1016
Page 518 of 2154

VIDEO 754 The DLR Hand
available from http://handbookofrobotics.org/view-chapter/19/videodetails/754
VIDEO 755 A high-speed Hand
available from http://handbookofrobotics.org/view-chapter/19/videodetails/755
VIDEO 756 The UBH2, University of Bologna Hand, ver. 2 (1992)
available from http://handbookofrobotics.org/view-chapter/19/videodetails/756
VIDEO 767 The Dexmart Hand
available from http://handbookofrobotics.org/view-chapter/19/videodetails/767
VIDEO 768 DLR Hand
available from http://handbookofrobotics.org/view-chapter/19/videodetails/768
VIDEO 769 The DLR Hand performing several task
available from http://handbookofrobotics.org/view-chapter/19/videodetails/769

References

19.1 T. Okada: Object-handling system for manual industry, IEEE Trans. Syst. Man Cybern. **2**, 79–86 (1979)
19.2 K.S. Salisbury, B. Roth: Kinematics and force analysis of articulated mechanical hands, J. Mech. Transm. Actuation Des. **105**, 35–41 (1983)
19.3 S.C. Jacobsen, E.K. Iversen, D.F. Knutti, R.T. Johnsan, K.B. Biggers: Design of the Utah/MIT dexterous hand, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (1986)
19.4 J. Butterfass, G. Hirzinger, S. Knoch, H. Liu: DLR's Multisensory articulated Hand Part I: Hard- and software architecture, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (1999)
19.5 A. Albu-Schäffer, T. Bahls, M. Chalon, O. Eiberger, W. Friedl, R. Gruber, S. Haddadin, U. Hagn, R. Haslinger, H. Hoppner, S. Jorg, M. Nickl, A. Nothhelfer, F. Petit, J. Reill, N. Seitz, T. Wimbock, S. Wolf, T. Wusthoff, G. Hirzinger: The DLR hand arm system, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (2011)
19.6 C. Melchiorri, G. Vassura: Mechanical and control features of the University of Bologna hand version 2, Proc. IEEE/RSJ Int. Conf. Int. Robots Syst. (IROS), Rayleigh (1992) pp. 187–193
19.7 W.T. Townsend: MCB – Industrial robot feature article- Barrett Hand grasper, Ind. Robot **27**(3), 181–188 (2000)
19.8 H. Kawasaki, T. Komatsu, K. Uchiyama: Dexterous anthropomophic robot hand with distributed tactile sensor: Gifu hand II, IEEE/ASME Trans. Mechatronics **7**(3), 296–303 (2002)
19.9 T.J. Doll, H.J. Scneebeli: The Karlsruhe Hand, Prepr. IFAC Symp. Robot Control (SYROCO) (1988), pp. 37.1–37.6
19.10 M.G. Catalano, G. Grioli, E. Farnioli, A. Serio, C. Piazza, A. Bicchi: Adaptive synergies for the design and control of the Pisa/IIT SoftHand, Int. J. Robotics Res. **33**, 768–782 (2014)
19.11 N. Fukaya, S. Toyama, T. Asfour, R. Dillmann: Design of the TUAT/Karlsruhe humanoid hand, Robot. Syst. **3**, 1754–1759 (2000)
19.12 A. Bicchi, A. Marigo: Dexterous grippers: Putting nonholonomy to work for fine manipulation, Int. J. Robotics Res. **21**(5/6), 427–442 (2002)
19.13 M.C. Carrozza, C. Suppo, F. Sebastiani, B. Massa, F. Vecchi, R. Lazzarini, M.R. Cutkosky, P. Dario: The SPRING hand: Development of a self-adaptive prosthesis for restoring natural grasping, Auton. Robots **16**(2), 125–141 (2004)
19.14 J.L. Pons, E. Rocon, R. Ceres, D. Reynaerts, B. Saro, S. Levin, W. Van Moorleghem: The MANUS-HAND dextrous robotics upper limb prosthesis: Mechanical and manipulation aspects, Auton. Robots **16**(2), 143–163 (2004)
19.15 Bebionic Prosthetic Hand: RSLSteeper, Leeds, UK (2015) http://www.bebionic.com/
19.16 G. Berselli, M. Piccinini, G. Palli, G. Vassura: Engineering design of fluid-filled soft covers for robotic contact interfaces: Guidelines, nonlinear modeling, and experimental validation, IEEE Trans. Robotics **27**(3), 436–449 (2011)
19.17 T. Iberall, C.L. MacKenzie: Opposition space and human prehension. In: *Dexterous Robot Hands*, (Springer, New York 1990)
19.18 A. Bicchi: Hands for dexterous manipulation and robust grasping: A difficult road toward simplicity, IEEE Trans. Robotics Autom. **16**(6), 652–662 (2000)
19.19 M.R. Cutkosky: On grasp choice, grasp models, and the design of hands for manufacturing tasks, IEEE Trans. Robotics Autom. **5**(3), 269–279 (1989)
19.20 G. Berselli, M. Piccinini, G. Vassura: Comparative evaluation of the selective compliance in elastic joints for robotic structures, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (2011) pp. 4626–4631
19.21 L.U. Odhner, A.M. Dollar: Dexterous manipulation with underactuated elastic hands, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (2011)
19.22 J. Butterfass, M. Grebenstein, H. Liu, G. Hirzinger: DLR-Hand II: Next generation of a dextrous robot hand, Proc. IEEE Int. Conf. Robotics Autom. (ICRA), Seoul (2001)
19.23 C. Melchiorri, G. Vassura: Mechanical and control features of the UB hand version II, Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS) (1992)
19.24 R.O. Ambrose, H. Aldridge, R.S. Askew, R.R. Burridge, W. Bluethmann, M. Diftler, C. Lovchik, D. Magruder, F. Rehnmark: Robonaut: NASA's space humanoid, IEEE Intell. Syst. (2000)
19.25 L. Birglen, C.M. Gosselin: Kinetostatic analysis of underactuated fingers, IEEE Trans. Robotics Autom. **20**(2), 211–221 (2004)

Avidbots Ex1016
Page 519 of 2154

19.26 I. Yamano, T. Maeno: Five-fingered robot hand using ultrasonic motors and elastic elements, Proc. IEEE Int. Conf. Robotics Autom. (2005) pp. 2684–2689
19.27 Shadow Dexterous Hand: Shadow Robot Co. LTD., London (2015), http://www.shadowrobot.com/
19.28 M. Kaneko, M. Higashimori, R. Takenaka, A. Namiki, M. Ishikawa: The 100G capturing robot – too fast to see, Proc. 8th Int. Symp. Artif. Life Robotics (2003) pp. 291–296
19.29 G. Palli, C. Natale, C. May, C. Melchiorri, T. Würtz: Modeling and control of the twisted string actuation system, IEEE/ASME Trans. Mechatronics **18**(2), 664–673 (2013)
19.30 G. Palli, C. Melchiorri, G. Vassura, G. Berselli, S. Pirozzi, C. Natale, G. De Maria, C. May: Innovative technologies for the next generation of robotic hands, Springer Tracts Adv. Robotics **80**, 173–218 (2012)
19.31 G. Palli, S. Pirozzi: Force sensor based on discrete optoelectronic components and compliant frames, Sensors Actuators A **165**, 239–249 (2011)
19.32 G. Palli, S. Pirozzi: A miniaturized optical force sensor for tendon-driven mechatronic systems: Design and experimental evaluation, Mechatronics **22**(8), 1097–1111 (2012)
19.33 W. Paetsch, M. Kaneko: A three fingered multi-jointed gripper for experimental use, Proc. IEEE Int. Workshop Intell. Robots Syst. (IROS) (1990) pp. 853–858
19.34 H. Maekawa, K. Yokoi, K. Tanie, M. Kaneko, N. Kimura, N. Imamura: Development of a three-fingerd robot hand with stiffness control capability, Mechatronics **2**(5), 483–494 (1992)
19.35 A. Bicchi: A criterion for optimal design of multiaxis force sensors, J. Robotics Auton. Syst. **10**(4), 269–286 (1992)
19.36 A. Pugh: *Robot Sensors: Tactile and Non-Vision*, Vol. 2 (Springer, Berlin, Heidelberg 1986)
19.37 H.R. Nicholls, M.H. Lee: A survey of robot tactile sensing technology, Int. J. Robotics Res. **3**(3), 3–30 (1989)
19.38 W.T. Townsend, J.K. Salisbury: Mechanical bandwidth as a guideline to high-performance manipulator design, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (1989)
19.39 S.D. Eppinger, W.P. Seering: Three dynamic problems in robot force control, IEEE Trans. Robotics Autom. **8**(6), 751–758 (1992)
19.40 W.T. Townsend, J.K. Salisbury: The effect of Coulomb friction and stiction on force control, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (1987)
19.41 G. Palli, G. Borghesan, C. Melchiorri: Modeling, identification and control of tendon-based actuation systems, IEEE Trans. Robotics **28**(2), 277–290 (2012)
19.42 M. Kaneko, T. Yamashita, K. Tanie: Basic considerations on transmission characteristics for tendon driven robots, Proc. 5th Int. Conf. Adv. Robotics (1991) pp. 827–883
19.43 R.M. Murray, Z. Li, S.S. Sastry: *A Mathematical Introduction to Robotic Manipulation* (CRC, Boca Raton 1994)
19.44 J. Mason, J.K. Salisbury: *Robot Hands and the Mechanics of Manipulation* (MIT Press, Cambridge 1985)
19.45 R.M. Cutkosky: *Robotic Grasping and Fine Manipulation* (Springer, New York 1985)

Avidbots Ex1016
Page 520 of 2154

Multimedia Contents

20. Snake-Like and Continuum Robots

Ian D. Walker, Howie Choset, Gregory S. Chirikjian

This chapter provides an overview of the state of the art of snake-like (backbones comprised of many small links) and continuum (continuous backbone) robots. The history of each of these classes of robot is reviewed, focusing on key hardware developments. A review of the existing theory and algorithms for kinematics for both types of robot is presented, followed by a summary of modeling of locomotion for snake-like and continuum mechanisms.

20.1 Snake Robots – Short History 481
20.2 Continuum Robots – Short History 485
20.3 Snake-Like and Continuum Robot Modeling 487
20.4 Modeling of Locomotion for Snake-Like and Continuum Mechanisms 491
20.5 Conclusion and Extensions to Related Areas 492
Video-References 492
References 493

20.1 Snake Robots – Short History

The study of snake robots began in earnest with the pioneering work of Shiego Hirose, a professor at the Tokyo Institute of Technology. He called his snake robots these active chord mechanisms, or ACMs for short. The first successful locomotive snake was ACM III (Fig. 20.1a), which he developed between 1972 and 1975; the ACM III robot had 20 segments and used passive wheels to form a constraint between the mechanism and the environment. Each segment was able to yaw with respect to each other and the combined motion of these degrees of freedom propelled the entire robot forward [20.1]. This type of coordinated control resulting in locomotion is called a *gait*; more formally, a gait is a cyclic motion in the mechanisms internal degrees of freedom that result in net motion for the snake robot. The next step in the development of the Hirose family of snake robots were the OBLIX (Fig. 20.1b) (1978–1979) and MOGURA (1982–1984), two robots that could achieve three-dimensional movements and lift their heads by rotating their oblique joints. The benefit of a single actuator operating an oblique joint over a true three-dimensional joint is simplicity, light weight, and ease to control because each segment only needed to be actuated by one motor [20.2].

Hirose's lab then began exploring possible applications of snake robots with Koryu I and II (1985–present). Each one of Koryu's load-bearing modules uses actuated joints to lift itself, lower itself, and in more recent incarnations, angle its wheel base to fit the ground. With such forms of actuations, these large snake robots were able to traverse uneven terrain and even climb stairs, all while carrying a load, distributed along the entire robot [20.2]. Inspired by the ability of real snakes to traverse uneven terrains, Hirose's lab developed a smaller mechanism, called Souryu I and II (1975–present) for the purposes of search and rescue (Fig. 20.2). The Souryu robots were able to climb over rubble because of their low center of mass and treads all along their bodies [20.3, 4].

In 1999, *Hirose* returned to the ACM with Slim Slime, a robot that was pneumatically actuated and formed a smooth curve instead of the usual sectioned, angular curve of other snake robots [20.5]. After Slim Slime, the lab began working on ACM-RI, R2, R3, and R4 (2001–present.) These robots are much closer to the design of the first ACM, with passive wheels and many-sectioned bodies. These robots, unlike the first ACM, can lift their bodies and move in three dimensions. Each

Avidbots Ex1016
Page 521 of 2154

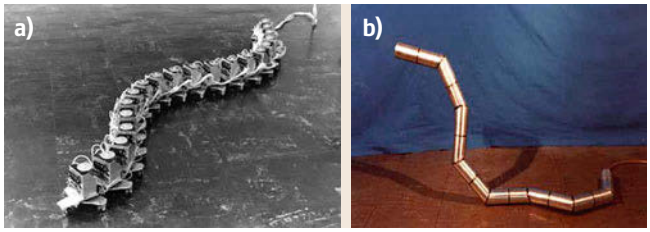


Fig. 20.1 Hirose's snake robots starting **(a)** with the active chord (ACM III) and **(b)** Oblix/Mogura mechanisms (after [20.2])

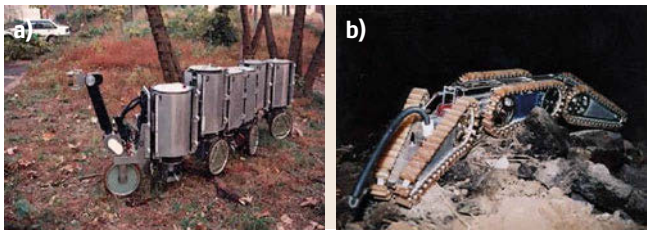


Fig.20.2a,b The Koryu I, II robot drving outside **(a)** (after [20.2]) and **(b)** the Souryu I, II robot maneuvering over rubble (after [20.4])

section has wheels that are set 90° from each other in an alternating fashion. They were created for use in search and rescue [20.1]. The most recent ACM is R5, a fully amphibious robot that uses passive wheels to roll along the ground and paddles to move through the water [20.6] (Fig. 20.3).

Snake robot research began in the United States in the late 1980s. Joel Burdick and his then-student Greg Chirikjian at Caltech developed a continuum theory to model the shape of snake robots in general. Burdick and Chirikjian coined the term *hyper-redundant manipulator*, which means a mechanism that has many more degrees of freedom that are necessary to position and orient its end-effector. They built Snakey in 1992; Snakey consisted of three degree of freedom bays or actuated trusses that could both manipulate and locomote. Their focus was to create a strong, precise mechanism that focused less on snake-like movement and more on the difficulties of programming a mechanism with many degrees of freedom. The Caltech team considered applications such as retrieval of satellites and search and rescue [20.7, 8]. Burdick and his group also designed and built snake robots with the Jet Propulsion Laboratory and the NEC Corporation with Nobuaki Takanashi. These were based on a three-dimensional joint prototype built by Takanashi. This smaller snake was able to execute several movement gaits, and was intended to be used in search and rescue.

Chirikjian moved on to Johns Hopkins University where he created a comprehensive research program on a broad class of hyper-redundant mechanisms, not limited to snake robots. One such nonsnake robot was the metamorphic shape-changing modular robot system comprising hexagonal elements. His group also developed snake robots based on binary actuators, which have two states, say fully elongated and contracted [20.9].

Snake and reconfigurable modular robot research also co-developed at Xerox Parc, and then at the University of Pennsylvania under the direction of Mark Yim. Beginning with PolyPod [20.10] in 1994 as part of Mark Yim's dissertation while he was at Stanford, snake configurations of modular robots were performing many of the conventional snake gaits as well as some innovative slinky-like and rolling gaits [20.11]. This research continued after Yim's move to Xerox's Palo Alto Research with PolyBot in 2000, a more robust version of PolyPod with more abilities for sensing [20.12]. When *Yim* began Modlab at the University of Pennsylvania, PolyBot evolved into CKbot ([20.13], 2009). CKbot can now reconfigure from an exploded position, run on four legs, walk on two, and perform most basic snake gaits [20.14, 15]. They also developed foamBot (2011), a robot that can create other robots, including snake robots, out of CKbot modules and hardening spray foam [20.16].

Hirose, Chirikjian, Burdick, and Yim had a strong influence on Howie Choset's work at Carnegie Mellon University (CMU). Choset's snake robot research began in the CMU Biorobotics Lab as an undergraduate research project which adapted Yim's modular design specifically for snake robot use (Fig. 20.7). Iterating on this modular design, this lab has created a series of snake robots over 15 years [20.17, 18]. In

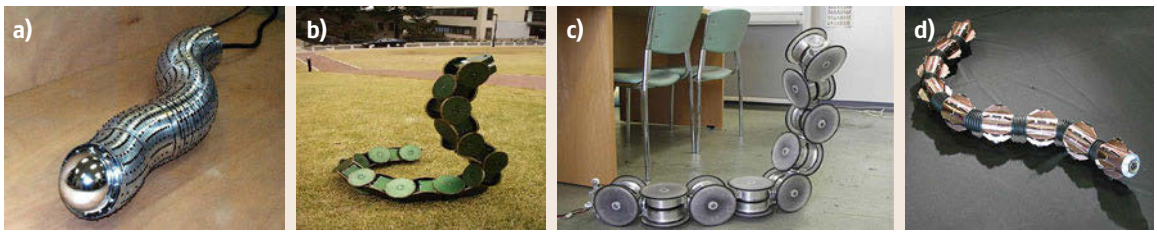


Fig.20.3a–d Slim Slime **(a)** and the new generation of ACM-R snake robots: **(b)** ACM-R3; **(c)** ACM-R4; **(d)** ACM-R5 (after [20.1])

Avidbots Ex1016
Page 522 of 2154



Fig. 20.4 Snakey, the snake robot showing also Joel Burdick, Greg Chirikjian, Howie Choset, Jimbo Ostrowski (after [20.8])

addition to the mechanism, one of the major innovations of this work is gaits, the controllers that prescribe motion for the snake robots. These gaits generalized the gait motion, based on Hirose's serpenoid curve and Chirikjian's modal functions, to perform a large variety of behaviors including climbing up poles, swimming in a pond, maneuvering through pipes, and breaching through a fence [20.19]. Working with Robin Murphy's group at Texas A&M, the snake robots have been fine tuned for urban search and rescue use. This created an ideal platform for infrastructure inspection such as networks of pipes in power plants. In 2012, Choset sent his robots to Egypt for archaeology in confined spaces [20.20].

In Norway, Øyvind Stavdahl in the Gemini Centre for Advanced Robotics created several snake robots, many of which were in collaboration with the Norwegian University of Science and Technology. The first of these major research projects in snake robotics was the Anna Konda (2005-present), a firefighting snake concept. Based on hydraulic actuation, this robot is powered entirely by the pressure of a fire hose, and is designed to reach flames in areas that are either too dangerous or too small for firefighters to reach [20.21].

This team also created AIKO (2006–2010), a snake robot created to experiment with snake locomotion that could later be applied to Anna Konda [20.22]. Mostly focused on object-aided locomotion, AIKO evolved into KULKO (2010–present), a robot with a force-sensing body that uses its simple understanding of the obstacles it is in contact with in order to locomote more effectively. These two robots have produced several innovative movement algorithms for object aided locomotion that could be applied to Anna Konda as well as other nonwheeled snake robots [20.23].

The snake robots described above rely on internal motions to propel themselves forward. *Johann Borenstein* at the University of Michigan created the Omni-Tread (2004) robot [20.24] that propels itself with moving threads situated along side of the robot. Its strong joints are actuated by pneumatic bellows. The bellows can be proportionally controlled, allowing its body to be fully compliant while crossing rubble to have more of its body touching the ground, or strong enough to lift half of its body weight, which is necessary for ledges and gaps. The Omni-Tread excelled at moving over rubble [20.24].

David Anhalt and James McKenna at SAIC took the tread concept to the limit; they designed and built a toroidal skin drive, a drive mechanism that has reciprocating skin that consists of an elongated toroidal skin that covers the entire length of the robot, and a drive unit which propels it. The outer (tubular) layer of the skin slides axially from the head to the tail of the snake robot, and wraps inside itself over a captured ring at the tail. The skin then recirculates from the tail to the head (through the center of the outer tubular layer), and changes direction again (over a second captured ring at the head) to again become the outside layer. In this configuration, the skin forms a continuous toroidal loop. The SAIC team collaborated with Choset's group to build a snake robot using this drive mechanism [20.25].

Finally, we describe some snake robots that use wheels to locomote. Hisashi Date, who heads a research team at the Robotics Lab for National Defense Academy of Japan, built a snake robot in 2007 to trace lines using lateral undulations and a line-tracing sensor arm. Date later developed in 2009 a mechanism that has passive wheels; it is very similar to Shigeo Hirose's first ACM robots, except it is powered by hydraulics. Date's robot also implements touch sensing much like being planned for Anna Konda [20.26].

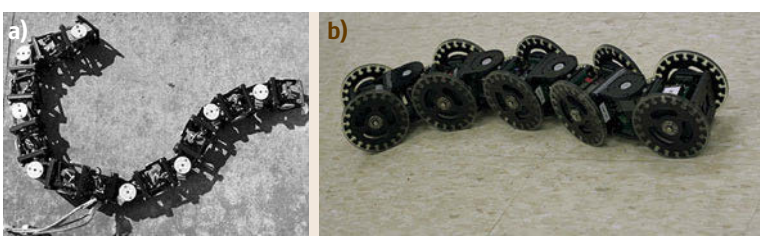


Fig. 20.5 **(a)** Yim's original Polypod robot (after [20.10]) and **(b)** current CK Robot in a wheeled snake configuration (after [20.13])

Avidbots Ex1016
Page 523 of 2154



Fig. 20.6 Choset's Modsnake consisting of 16 DOFs (degree of freedom) each orthogonally placed with respect to each other (after [20.17, 18])



Fig. 20.7 Anna Konda snake robot (after [20.21])

Another wheeled snake robot is HAUBOT (2009-present), developed by Akio Ishiguro at Tohoku University. It is being used to study the autonomous decentralized control mechanisms underlying the adaptive locomotion of animals. It uses real-time tunable rubber springs to actuate each joint. Each joint generates torque proportional to the difference between the joint angle and the target angle which is given by the anterior joint angle. By using just this method of simple sensing, the robot can locomote in two dimensions, without requiring a centralized control hub. More recently, HAUBOT has been successfully paired with head tracking software so that it can be driven intuitively [20.27].

At Kyoto University, Fumitoshi Matsuno has also developed several snake robots. Matsuno has made many innovations in the control of snakes with different modes of propulsion. In 2002, Matsuno and his lab built KOHGA, a treaded snake robot made for search and rescue. Its joints can be passive or active, allowing it to move through rough terrain and also lift parts of its body [20.28]. Another robot they have developed is a snake robot that propels itself using screw-inspired modules. Built in 2010, it uses many of the same control algorithms as KOHGA except besides moving forward, backward, and turning, it can also roll sideways perpendicular to its head direction [20.29].

These labs together have created an incredible variety of snake-like robot locomotion. Snake robots can now traverse nonflat fields, climb poles, swim underwater, and even rebuild themselves. They can access incredibly small spaces and transition between swimming to slithering to climbing without having to be reconfigured. Advancing far beyond slithering across flat ground, these robots are nearly ready to begin tackling spaces in the field that no other robot can reach.

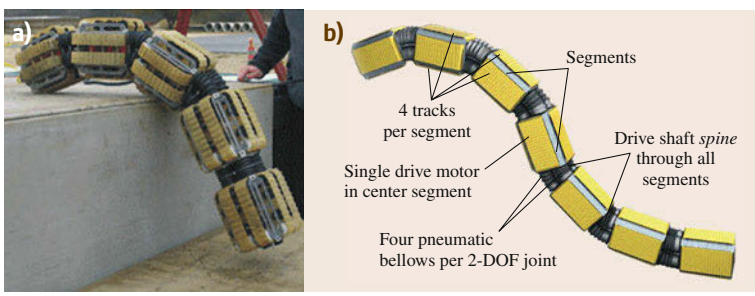


Fig.20.8a,b The Omni-Tread treaded snake robot (a) and an (b) illustration of its design (after [20.24])

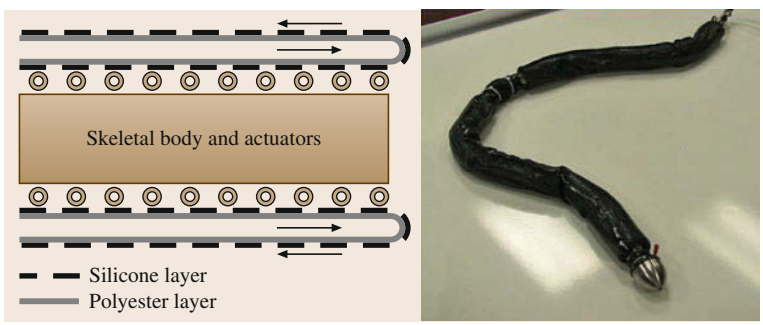


Fig. 20.9 The SAIC skin drive mechanism followed by the SAIC/CMU snake robot powered by the skin drive mechanism (after [20.25])

Soon these robots may be found in the pipes of oil refineries, crawling through nuclear power plants, at the front line of search and rescue operations, fighting fires or exploring underwater environments.

20.2 Continuum Robots – Short History

The term *continuum manipulator*, first coined by *Robinson* and *Davies* in [20.30], which is equivalent to the *continuous morphology manipulator* defined in [20.31], indicates the snake/hyper-redundant robot concept taken to the extreme. In other words, conceptually this category of robots have backbone structures *taken to the limit*, with their number of *joints* tending to infinity, but with their *link lengths* tending to zero, which is also the approach taken in Chirikjian and Burdick's work on backbone curves. While this concept initially appears idealistic, complicated and impossible to implement, the limiting case (a smooth *continuum* curve, with the ability to bend at any point along the backbone) is quite easy to realize in hardware.

Numerous continuum designs have been proposed since the early 1960s beginning with the *Tensor Arm* [20.32] developed based on the original *Orm* concept of Leifer and Scheinman [20.33]. With a flexible backbone bent by remotely driven tendons, the Tensor Arm anticipated numerous later designs. However – a key issue with later continuum robot designs until the 1990s – coordination of the inputs to take advantage of the shapes achievable by the robot proved difficult. This lack of enabling underlying models for motion planning (now much less of an issue, see the following section) hindered the development of the field for many years.

Nevertheless, some researchers, beginning with *Shigeo Hirose* [20.2], continued to explore the problem of constructing continuum robots. Many of these efforts were inspired by an evolving knowledge of the morphology and functionality of biological continuum structures [20.34–36] particularly the trunks of elephants [20.37–39] and the arms of octopuses [20.40–42]. The key robot design issues revolve around how to actuate (bend and possibly extend/contract) the backbone. Two basic design strategies have emerged [20.43, 44]: (1) extrinsic actuation, where the actuators are separate from the backbone structure, e.g., [20.45–49]; and (2) intrinsic actuation, where the actuators represent a fundamental part of the backbone, e.g., [20.50–53].

Extrinsically actuated designs have the advantage that by locating the actuators outside the robot workspace, the backbone itself can be simplified and streamlined. Tendons have proved a popular and generally successful choice for extrinsically actuated continuum designs [20.38, 54, 55]. Groups of tendons are terminated at specific points down the backbone, and combine to bend the backbone to produce a series of (usually constant curvature) *sections*. The tendons bend a backbone that is nominally straight (realized either via an elastic rod [20.56], the incorporation of springs [20.38], or pneumatic chambers [20.57, 58]). A good example of the ability of this design strategy to produce a long, thin backbone with extrinsic actuation is given by NASA Johnson Space Center's Tendril ([20.49], Fig. 20.10), intended to penetrate through small orifices and explore in tight spaces for Space applications.

In recent years, a completely different form of extrinsically actuated continuum robot, based on concentric tubes, has appeared. These concentric tube [20.59, 60] (sometimes termed as active cannula [20.61]) robots have been primarily motivated by medical applications [20.62]. Research has focused on application (e.g., as active endoscopes [20.63–66]) within the human body [20.67, 68], on various types of minimally invasive surgical procedures [20.69–71], and the associated development of *steerable needle* technologies [20.72–74].

Concentric tube robots, as the name suggests, form the robot backbone from several (typically two or three) hollow tubes, with tubes of smaller radius inserted through those of larger radius. Each smaller tube can be pushed through the tube immediately outside it to extend the backbone length. The tubes can also be rotated, to give $2n$ degrees of freedom for an n-tube backbone. The tubes are preformed to have a constant curvature, so can be extended and bent (albeit with amount

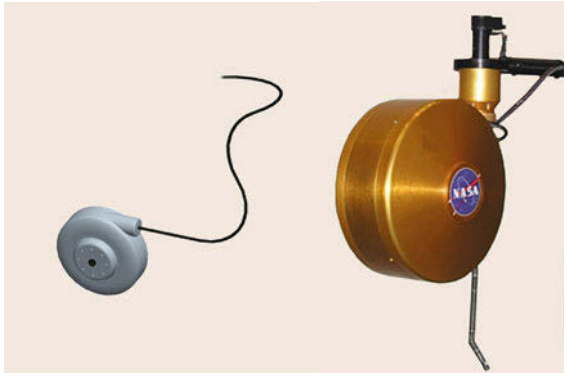


Fig. 20.10 NASA's Tendril, an extrinsically actuated continuum manipulator (after [20.49])

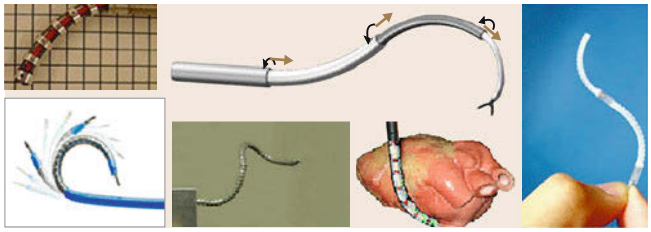


Fig. 20.11 Extrinsically actuated concentric tube continuum manipulators



Fig. 20.12 The OC Robotics *Snake-Arm* robot – an extrinsically actuated *continuum-like* robot with segmented backbone (after [20.75])



Fig. 20.13 The intrinsically actuated *Octarm* pneumatic continuum manipulator and its development team (after [20.42])

of bending preset) in arbitrary directions (Fig. 20.11). This design arrangement is well-suited to the medical applications which have motivated its development. Numerous prototypes have been developed in the last several years, and the results of initial trials are highly promising. This work is being pioneered by the groups of *Pierre Dupont* at Boston University [20.59, 73, 74, 76], *Michael Zinn* et al. at the University of Wisconsin [20.62, 77], and *Robert Webster*, III et al., at Vanderbilt University [20.44, 61, 67, 78].

Note that the backbone for extrinsically actuated robots does not have to be continuous to produce the continuum effect [20.47, 79, 80]. A commercially successful group of extrinsically actuated continuum-like *snake-arm* manipulators are currently manufactured and retailed by OC Robotics (OCR) in the United Kingdom [20.75]. Remotely tendon-driven (designed by Rob Buckingham and Andy Graham of OCR), but with a snake-like segmented backbone (the number and form of segments can be changed) these snake-arms give a continuous backbone effect. OCR's manipulators (Fig. 20.12) are currently the only general purpose continuum-like robot commercially available, and have been deployed in nuclear reactors and inside airframes, among other applications for largely noncontact operations in tight, cluttered spaces. This ability to penetrate and *slither through* complex environments (mirroring NASA's goals for Tendril) is one of the main motivations and anticipated practical advantages of continuum robots.

Intrinsically actuated continuum robots have their actuators within the continuum backbone. Indeed, the backbones of most intrinsically actuated continuum robots are largely formed by their actuators. Artificial muscle technologies, notably pneumatic McKibben muscles, have proved most effective in intrinsically actuated hardware realizations [20.50, 81]. A particular advantage of intrinsic actuation is the ability to produce extensible backbones [20.43, 44].

For example, Ian Walker's group at Clemson led the development of the octopus-arm inspired *Octarm* instrinsically actuated extensible continuum manipulators ([20.42], Fig. 20.13, VIDEO 158). The backbone of the Octarm is formed from its actuators: sets of (3) McKibben muscles arranged in series to give three or four sections. Several generations of Octarm were tested in field trials, mounted on a Foster-Miller Talon mobile base [20.81] (VIDEO 157). The robots were shown to be able to grasp and manipulate objects over a wide range (order of magnitude) of sizes and weights, and of widely varying shapes and textures [20.82]. This

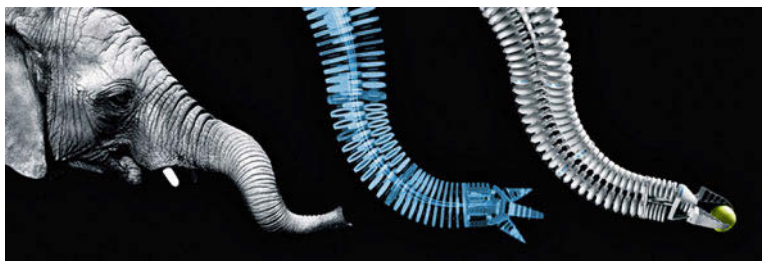


Fig. 20.14 The intrinsically actuated bionic handling assistant by *Festo* (after [20.50])

ability to adapt shape to perform *whole-arm manipulation* [20.83] with the backbone is another prime motivation for continuum robots [20.84].

Another good example of an intrinsically actuated continuum robot is the Bionic Handling Assistant recently developed by *Festo* [20.50] (Fig. 20.14). This continuum robot is unusual in that it is preshaped for its sections to be nonconstant curvature. The curvature is biologically inspired.

Currently, another biologically inspired (once again by the octopus) large project, under EU funding, led by Darwin Caldwell, and featuring researchers in Greece, Israel, Italy, Switzerland, and the UK, is working toward creation of a *European Octopus* robot [20.40, 85] (Fig. 20.15). This robot features multiple continuum limbs [20.85]. The European group has considered both intrinsic [20.86] and extrinsic actuation [20.40] for their limbs.

Many snakelike/continuum robotic devices have found applications in the medical area. This includes very flexible steerable needles [20.87–91], and the concentric-tube robots mentioned earlier [20.73, 74, 92–97]. Other recent medical snake robots include novel articulated designs [20.98, 99], and stiff-continuum-filament cable-driven designs [20.100–103].

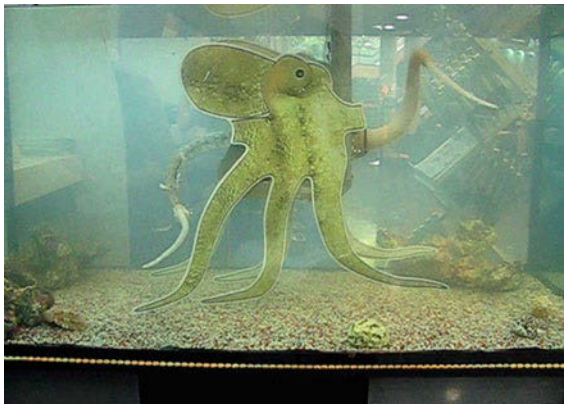


Fig. 20.15 The *European Octopus* continuum-limbed robot (after [20.40]) (currently under development)

20.3 Snake-Like and Continuum Robot Modeling

While continuum manipulators are seen to be the extreme case of snake-like or hyper-redundant structures, it was quickly realized that they represent a fundamentally new class of manipulators [20.43, 44, 104]. A continuous flexible backbone features, at least in theory, an infinite number of degrees of freedom. As such, the traditional tools in robotics for modeling (a finite number of) serial rigid links no longer apply. Additionally, it is obviously not possible to actuate an infinite number of degrees of freedom in practice. Continuum manipulator hardware universally features a finite number of actuators, applying forces/torques to the backbone at a fixed and preselected set of locations. This causes significant complexity in their analysis. However, significant progress has been made, and understanding resulting from analysis of continuum kinematics has been applied, as discussed in the following section.

Clearly, to correctly model continuum manipulators, models based on continuous backbones are required. Interestingly, such models have also proved to be key theoretical resources in the motion planning of rigid-link hyper-redundant systems. While in theory discrete-link hyper-redundant manipulators can be modeled (and their planning based on) traditional manipulator modeling techniques, it was quickly found that the computational complexity of such approaches made them difficult to implement, and the corresponding models hard to visualize. An alternative strategy (and a more successful one in practice) has been to adopt an approach based on continuum manipulator kinematics. In the following, we overview the recent development of continuum manipulator kinematics. We then discuss how the existence of both discrete-link and continuum models have enhanced motion planning for each category of hyper-redundant system.

The lack of distinct links in continuum systems makes the standard robot manipulator modeling strategy of a finite number of coordinate frames (each fixed in one link) inappropriate for their modeling. Instead, the natural approach is to model continuum kinematics via a frame which evolves along a continuous backbone, parameterized by arc length s. Local motion of the backbone at point s is modeled in terms of the local frame at s. This strategy allows computation of *forward kinematics*, and construction of *continuum Jacobians*, analogous to those for rigid-link systems [20.31, 105].

Numerous alternatives for selection of the backbone frames have been proposed [20.31, 106–109]. In all such approaches, positions along the backbone curve can be expressed relative to the unit tangent, $u(s)$, as

$$x(t) = \int_0^t [1 + \varepsilon(s)]u(s)\,\mathrm{d}s\,.$$

Avidbots Ex1016
Page 527 of 2154

In this formulation, $\varepsilon(s)$ is the local extensibility, or stretch, of the curve. If this quantity is equal to zero, then t becomes arc length. Otherwise, equal increments of t can correspond to different lengths along the curve, with a positive value of $\varepsilon(s)$ corresponding to a stretch, and negative values corresponding to a compression. While most continuum robots have a backbone that is essentially inextensible, the reason for introducing this variability is that allowing certain sections to stretch or compress affords greater freedom to hyper-redundant manipulators with discrete stages, making it easier to *fit* the physical robot to the curve. This fitting process takes as an input not only the relative positions of the distal and proximal ends of each stage, but also orientations. If the tangent is taken as one of the axes of a reference frame, a choice for the other two axes remains and can be made in a number of different ways. Once this choice is made, a reference frame

$$g(t) = (R(t), x(t))$$

is established at each value of t and relative rigid-body displacements are computed for stages as

$$[g(t_i)]^{-1} \circ g(t_{i+1}) = \left([R(t_i)]^{\mathrm{T}} R(t_{i+1}), [R(t_i)]^{\mathrm{T}} (x(t_{i+1}) - x(t_i))\right) .$$

In the inextensible case where the curve parameter becomes arclength s one natural choice for defining the orientation $R(t)$ is the well-known Serret–Frenet frame [20.31, 108, 109] which evolves along the backbone according to the following

$$\begin{aligned} \mathrm{d}t/\mathrm{d}s &= \kappa n \\ \mathrm{d}n/\mathrm{d}s &= -\kappa t + \tau b \\ \mathrm{d}b/\mathrm{d}s &= -\tau n \, . \end{aligned}$$

In the above, the frame origin is given by x, where the unit tangent to the curve $t = \mathrm{d}x/\mathrm{d}s$ (denoted as t rather than u), and forms one of the frame axes. The other axes are defined as the normal ($t \cdot n = 0$) and the binormal $b = t \times n$. The curvature κ and torsion τ dictate the shape of the curve. The axes of the Frenet–Serret frame provide an intuitive visualization of local movement: two possible dimensions of bending, corresponding to rotations about the normal and binormal axes, and one of extension/contraction (present and controllable in several continuum robots), corresponding to translation along the tangent axis.

The Frenet–Serret model has the advantage of being a well-established means for modeling continuous spatial curves. The drawback of this formulation is that it is not *optimal* in the sense that for a given curve shape, the total amount of motion of the evolving reference frame required to traverse the curve is not minimal subject to end constraints and the requirement that one axis of the reference frame is the tangent. Variational calculus can be used to obtain the optimal framing of curves, as reviewed in [20.31] in the context of coordinate-dependent optimization, and in [20.110] using a coordinate-free approach.

Moreover, there are multiple approaches to defining the evolution of the tangent, and hence the shape of the curve. Consider an inextensible curve whose shape has not yet been specified, and let the unit tangent to the curve be defined relative to an a priori-unknown arclength-dependent orientation as

$$u(s) = R(s) e_3 \, .$$

The goal is to find $R(s)$ such that the positional constraint

$$x(L) = \int_0^L u(s) \, \mathrm{d}s \, ,$$

and the orientational constraints on $R(L)$ are satisfied when $R(0) = I$. Here, I is the identity matrix and the manipulator has length L.

Treating s as time, then corresponding to $R(s)$ is an angular velocity $\omega = (R^{\mathrm{T}} \, \mathrm{d}R/\mathrm{d}s)^V$, where the V operation extracts the dual vector of a skew-symmetric matrix. We can then ask what $R(s)$ is optimal in the sense of minimizing a cost of the form

$$I = \frac{1}{2} \int_0^L [\omega(s) - b]^{\mathrm{T}} B [\omega(s) - b] \, \mathrm{d}s \, .$$

When $b = 0$ and B is the moment of inertia of a cross section of the manipulator, this is akin to kinetic energy of rotation, and minimizing it provides an evolution of reference frames that minimally rotate. Variational calculus provides the conditions

$$B\dot{\omega} + \omega \times (B\omega - b) = \begin{pmatrix} -\lambda^{\mathrm{T}} R e_2 \\ \lambda^{\mathrm{T}} R e_1 \\ 0 \end{pmatrix} ,$$

that can be integrated numerically and the rotation matrix $R(s)$ is obtained from the angular velocities by integrating

$$\dot{R} = R \left[\sum_{i=1}^{3} \omega_i(s) E_i \right] ,$$

Avidbots Ex1016
Page 528 of 2154

where

$$E_1 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{pmatrix} ,$$

$$E_2 = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{pmatrix} ,$$

$$E_3 = \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} .$$

The three-dimensional vector of Lagrange multipliers λ and the a priori unspecified initial value of the three-dimensional vector ω at $s = 0$ together provide six degrees of freedom with which to match the end constraints on position and orientation. In this way, curves of fixed length that vary in shape as little as possible while meeting end constraints can be generated.

Alternative frame representations in the continuum robot literature also have selected the axes of the frame to align with the controlled motion axes of particular hardware realizations [20.47, 107]. In either case, given a frame, the key problem is how to use the resulting model to plan motions. The basic issue is that the continuum kinematics models in their basic form feature infinite degrees of freedom (as necessary to model arbitrary spatial curves). However, hyper-redundant robots (discrete-joint or continuum) can be controlled in only a finite number of ways, thus admitting a reduced set of physical solutions. Therefore, research in continuum manipulators has concentrated on how to constrain continuum kinematics models to best represent robot hardware.

The initial major breakthrough in motion planning for hyper-redundant arms was made in a landmark series of publications by *Chirikjian* and *Burdick* [20.31, 105, 111] introducing continuum kinematics and using them to approximate rigid-link hyper-redundant systems [20.105]. Basically, the philosophy is to use a (theoretical) curve to model the physical backbone of a hyper-redundant robot. Motion planning is performed for the curve, and the (discrete) robot backbone then *fit* to the resulting (continuous) solution curve. This approach has proved quite effective, and the associated research introduced several key theoretical concepts to the area. In particular, the use of a modal approach [20.105, 112] (restricting the classes of allowable solutions to shapes generated via simple linear combinations of *modes*), arose from this approach to redundancy resolution for hyper-redundant manipulators. This concept can be viewed as a *top-down* approach of building a general model, and adapting it (via mode selection and curve-fitting) to hardware.

In the planar case, the angle of the rotation matrix $R(s)$ is simply the integral of the curvature of the curve. Expressing this curvature as a weighted sum of two functions, or modes, then restricts the manipulator to act as if it only has two degrees of freedom. As an example of this modal approach, a closed form solution can be obtained for inextensible manipulators of unit length by restricting the curvature to be a linear combination of $\cos 2\pi s$ and $\sin 2\pi s$. If the coefficients in front of these functions are, respectively, $2\pi a_1$ and $2\pi a_2$, then

$$x_{ee} = \sin(a_2) J_0 \left[(a_1^2 + a_2^2)^{\frac{1}{2}} \right] ,$$

$$y_{ee} = \cos(a_2) J_0 \left[(a_1^2 + a_2^2)^{\frac{1}{2}} \right] ,$$

where J_0 is the zeroth-order Bessel function.

And inverse kinematics can be achieved in closed form by computing

$$\begin{aligned} a_1 &= \hat{a}_1^{\pm}(\bar{x}_{ee}) \\ &= \pm \left(\left\{ J_0^{-1} \left[(x_{ee}^2 + y_{ee}^2)^{\frac{1}{2}} \right] \right\}^2 \right. \\ &\quad \left. - [\mathrm{Atan2}(x_{ee}, y_{ee})]^2 \right)^{\frac{1}{2}} \\ a_2 &= \hat{a}_2(\bar{x}_{ee}) = \mathrm{Atan2}(x_{ee}, y_{ee}) . \end{aligned}$$

In this planar example the convention that the tangent to the curve at the base of the manipulator points along the y axis has been used.

This closed-form solution has also been used to generate closed-form solutions for spatial manipulators, as well as to define waves that travel along the manipulator for the purpose of obstacle avoidance, locomotion and manipulation. This is illustrated in Figs. 20.16–20.18 [20.31] with a variable-geometry truss superimposed on the backbone curve. In the context of obstacle avoidance, the modal solution given above is scaled in order that the part of the manipulator outside of the obstacle field terminates at o position to allow entry, and the section inside of the obstacle field has curvature that is piecewise constant to maneuver around the obstacles.

In the context of locomotion, the modal approach is used to ensure that the end conditions on a traveling wave match those of terrain. In the context of grasping

Avidbots Ex1016
Page 529 of 2154

and manipulation, the modal approach is used in two ways. First, it defines the shape of a locomotion-like traveling wave that traverses the grasped object. This extends the reach of the manipulator around the object. Then, the bottom part of the manipulator contracts to form a shape defined by the modes while the part of the manipulator that is closest to the base yet in contact with

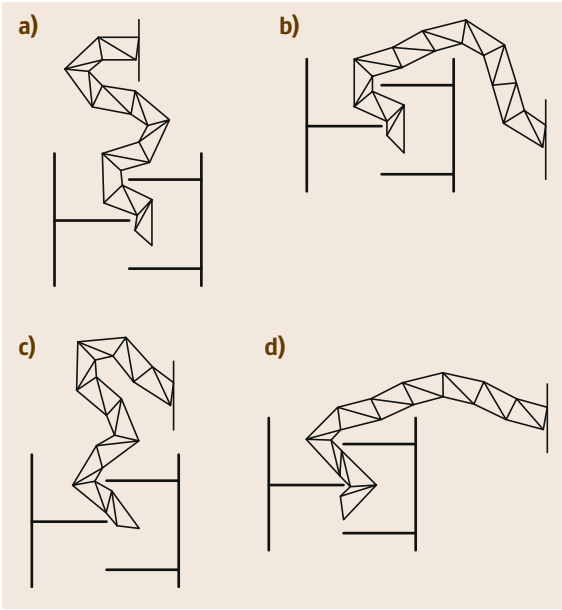


Fig.20.16a–d Using the modal approach to position part of a manipulator at the entry of an obstacle field (after [20.31])

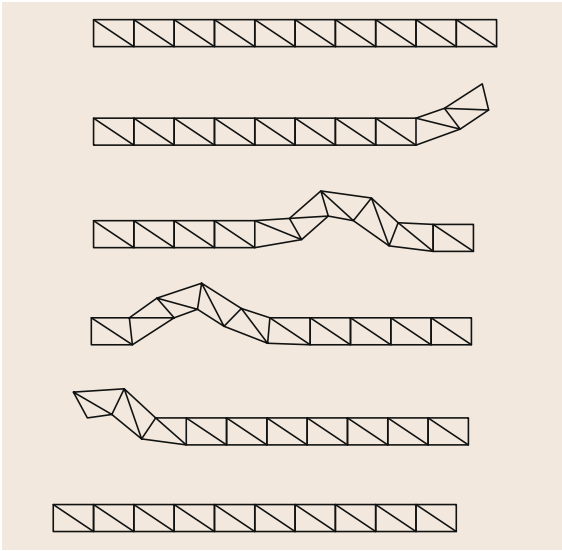


Fig. 20.17 Net motion to the left by wave propagation (after [20.31])

the grasped object unwinds. The result is a combination of waves and unwinding that produce net rotation of grasped objects without ever letting go, as described in [20.31, 113].

More recent research has concentrated on the notion *dual* to that of the previous section, using the physical constraints imposed on the backbones of specific continuum robot hardware types to construct continuum kinematics. This can be viewed as a *bottom-up* approach, focused on particular hardware classes, with the main aim of sufficiently modeling the hardware to avoid the use of approximations in the motion planning. Examples of this approach are given in [20.44, 107] where the key constraint imposed on a general continuum kinematics model arises from the observation that many continuum hardware implementations resolve into a finite number of constant curvature sections. (This arises naturally from the imposition of a finite number of input force/torques on a stiff continuum backbone). In [20.106], it is noted that the resulting constant curvature models can be viewed as a particular case of the general modal approach of [20.105].

The approach shown [20.44, 107] involves several transformations [20.114], to transition the map between task and actuator space via a *section* space. As key part of this transformation, a conventional (theoretical) rigid-link model can be used to model the kinematics of each section. Thus, rigid-link kinematics have proven key to resolving redundancy for continuum manipulators, as continuum kinematics play a key part in redundancy resolution for rigid-link hyperredundant arms.

Finally, a series of Jacobians are formed. The difference here is that the continuum Jacobian is a function of locally meaningful variables (bending angle, curvature, and extension) defining section shape or the direct values of actuators which determine these variables. Given the existence of continuum Jacobians, the redundancy is usually resolved as

$$\dot{\mathbf{l}} = J_E^{\dagger} \dot{l}_E + \left(I - J_E^{\dagger} J_E \right) \dot{\mathbf{l}}_0$$

in the same way as for conventional redundant manipulators (and with the same general corresponding advantages and issues), as discussed in Chap. 10.

Recent efforts have extended the progress in continuum robot kinematics to develop dynamic models for continuum robots [20.84, 104]. New models based on the Lagrangian [20.108, 115] and Newton–Euler [20.116] formulations have been introduced. Progress is impeded by the complexity of the dynamics of continuous backbones with large deformations, and many models remain too complex for practical

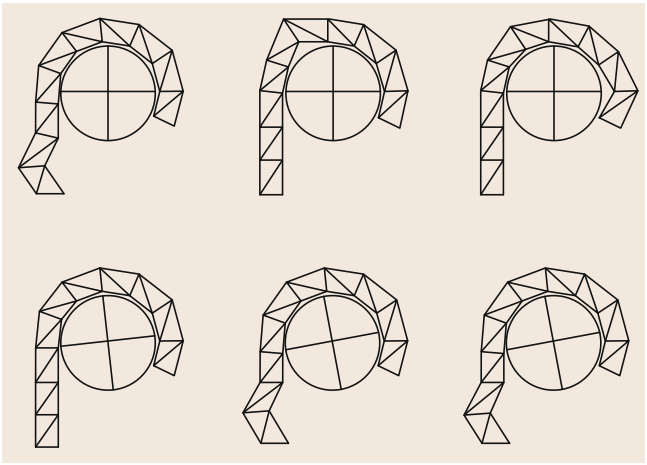


Fig. 20.18 Net rotation of an object by wave initiation at base and wave propagation (after [20.31]) ►

implementation. Hence, recent work has focused on implementable dynamic models based on simple, lumped-parameter models [20.117].

The new kinematic and dynamic models are in turn now enabling research in continuum robot motion planning [20.118, 119] and control [20.120, 121]. New models for force estimation [20.122], handling [20.76, 123], and sensing [20.124] using continuum robots have been introduced. Both model-based [20.54] and non-model based [20.125] controllers have been proposed. Controllers have been proposed in both configuration and task [20.54, 126] space. However, at this time, implementation of model-based controllers is rare.

20.4 Modeling of Locomotion for Snake-Like and Continuum Mechanisms

Snake and continuum robots have the ability to use their entire bodies to interact with the environment in order to produce purposeful motion [20.127–136]. Locomotion is a challenging motion planning and control problem because a planner/controller must contend with many degrees of freedom at once, even if they were somehow reduced with approaches like the backbone method, discussed above. Many researchers achieve this by modeling the snake robot's motion with a *gait*, a cyclic motion in the internal degrees of freedom of a mechanism that produces net motion. It makes sense to implement gaits because all locomoting animals exhibit gaits: horses gallop, trot, and walk; snakes sidewind, laterally undulate, and linearly progress.

The benefit of the gait is that one can reduce a complex system down to a few set of parameters that describe the gait and then by cycling through the phase of the gait, a planner can achieve locomotion. So, if the gait were just a sinusoid, the user specifies amplitude, frequency, and phase and the mechanism is then *fit* to the sinusoid, which requires very little computation as was seen with the backbone method. To achieve motion in three dimensions, we describe gaits for the snake robot with two sinusoidal waves lying in two mutually perpendicular planes, a horizontal plane parallel to the ground and a vertical plane perpendicular to the ground. Recall that some of the mechanisms, described above, consists of single degree of freedom joints, each placed orthogonally to its neighboring joints. For the sake of notation, we ascribe the even joints to modules that bend in the vertical plane and odd joints to modules that bend in the horizontal plane. So, if a vertical wave perpendicular to the ground is sent through the mechanism, only the even modules participate, while the odd modules remain stationary. Likewise, only the odd modules participate in sending lateral waves parallel to the ground through the robot. Should the snake robot fall onto its side, then even modules simply become the lateral modules and the odd become the vertical modules.

Given that the even modules are responsible for propagating vertical waves and the odd for propagating lateral waves, we use the terms $\text{offset}_{\text{vertical}}$, $\text{offset}_{\text{lateral}}$, $\text{amplitude}_{\text{vertical}}$, and $\text{amplitude}_{\text{ground}}$ to denote the parameters affecting the two orthogonal waves. Finally, we assume that both offset parameters are set to zero, unless otherwise noted. All of our differentiable gaits can be described by an angle of the n-th module at time t as

$$\begin{aligned}
&\text{angle}(n,t) \\
&\quad = \begin{cases} \text{offset}_{\text{even}} + \text{amplitude}_{\text{even}} * \sin(\theta)\,, \\ \quad n = \text{even}\,, \\ \text{offset}_{\text{odd}} + \text{amplitude}_{\text{odd}} * \sin(\theta + \delta)\,, \\ \quad n = \text{odd}\,, \end{cases}\,, \\
&\theta = \left(\frac{\mathrm{d}\theta}{\mathrm{d}n} * n + \frac{\mathrm{d}\theta}{\mathrm{d}t} * t \right)\,.
\end{aligned} \tag{20.1}$$

The offset is generally used to aim the robot when locomoting on the ground. For instance, when the snake is straight or climbing a pipe, the offset is zero. The am-

Avidbots Ex1016
Page 531 of 2154

plitude terms describe the amplitudes of the mutually perpendicular waves. In most cases, the amplitude and speed of the robot are directly correlated and as both increase, the robot can better reach over large obstacles. However, larger amplitude comes at the cost of reduced stability. Intuitively, given a fixed number of modules, the fewer waves in the robot, the larger each wave can be. Thus, smaller values of $d\theta/dn$ allow more modules to participate in any one wave, which in turn allows for larger waves.

The frequency of the sine wave is determined by θ, which increases with time and along the length of the snake robot. The rate at which θ changes is controlled by the $d\theta/dn$ and $d\theta/dt$ terms. Note that the sine wave moving through the robot has a spatial and temporal component: 1) $d\theta/dn$ determines the spatial component, meaning if we freeze time (fix t), it tells us what the robot as a whole looks like, and 2) $d\theta/dt$ determines the frequency of the temporal component, meaning that if we examine the sinusoidal trajectory of one individual actuator (fix n), we determine how it moves. A zero value for $d\theta/dn$ causes identical angles on all of the modules on one axis, generating an arc along that axis, and a zero value for $d\theta/dt$ has the effect of freezing time. Finally, δ is simply an offset to control the timing between the motions of the two orthogonal waves. These parameters can be modified to change the type of and nature of the gait performed by the robot.

Current work includes combining the gait-equation-based approach with backbone curves. We are developing techniques that take an animated sequence of backbone curves and infer the gait parameters from that sequence. The backbone curve approach is easy to visualize and therefore geometrically manipulate. The benefit of the gait approach is that it contains an analytic expression which permits evaluation, and hence optimization. With this new current approach, we can obtain the best of both worlds.

20.5 Conclusion and Extensions to Related Areas

The field of snake-like and continuum robots has blossomed over the past 40 years. The work has taken many different directions including novel designs for locomotion, manipulation, and medical applications, as well as different avenues for applying the resulting modeling techniques including nonholonomic planning and the conformational flexibility and statistical mechanics of biological macromolecules.

It is worth noting that other related works in nonrobotic fields and application areas have developed either in parallel or as a consequence of the efforts mentioned before. For example, continuum devices were developed essentially outside the field of robotics in [20.137, 138] from the perspective of mechanics. Continuum curve approaches applied to variable-geometry trusses were developed in [20.139–141], essentially in parallel with [20.9]. In recent work, wormlike robots [20.142] have been designed that execute peristaltic waves, motivated by biology akin to those described in [20.113]. Finally, analysis for understanding curves and workspace formulations for snake and continuum robots have been used to study deoxyribonucleic acid (DNA) [20.110, 143–148].

Video-References

VIDEO 157 Field experiments with the OctArm continuum manipulator
available from http://handbookofrobotics.org/view-chapter/20/videodetails/157

VIDEO 158 OctArms I-V
available from http://handbookofrobotics.org/view-chapter/20/videodetails/158

VIDEO 159 One dimensional binary manipulator
available from http://handbookofrobotics.org/view-chapter/20/videodetails/159

VIDEO 160 Two dimensional binary manipulator
available from http://handbookofrobotics.org/view-chapter/20/videodetails/160

VIDEO 161 Three dimensional binary manipulator
available from http://handbookofrobotics.org/view-chapter/20/videodetails/161

VIDEO 162 Binary manipulator grasping
available from http://handbookofrobotics.org/view-chapter/20/videodetails/162

VIDEO 163 Binary manipulator obstacle navigation
available from http://handbookofrobotics.org/view-chapter/20/videodetails/163

Avidbots Ex1016
Page 532 of 2154

VIDEO 164 Binary manipulator object recovery
available from http://handbookofrobotics.org/view-chapter/20/videodetails/164
VIDEO 165 Modsnake fence navigation
available from http://handbookofrobotics.org/view-chapter/20/videodetails/165
VIDEO 166 Modsnake autonomous pole-climbing
available from http://handbookofrobotics.org/view-chapter/20/videodetails/166
VIDEO 167 Modsnake pipe inspection
available from http://handbookofrobotics.org/view-chapter/20/videodetails/167
VIDEO 168 Modsnake climbing a tree
available from http://handbookofrobotics.org/view-chapter/20/videodetails/168
VIDEO 169 Modsnake swimming
available from http://handbookofrobotics.org/view-chapter/20/videodetails/169
VIDEO 171 Modsnake pole climb
available from http://handbookofrobotics.org/view-chapter/20/videodetails/171
VIDEO 174 Modsnake sidewinding
available from http://handbookofrobotics.org/view-chapter/20/videodetails/174
VIDEO 175 CMU medical snake robot
available from http://handbookofrobotics.org/view-chapter/20/videodetails/175
VIDEO 244 Active compliant insertion
available from http://handbookofrobotics.org/view-chapter/20/videodetails/244
VIDEO 245 Automatic insertion implant calibration
available from http://handbookofrobotics.org/view-chapter/20/videodetails/245
VIDEO 246 IREP tagging spikes
available from http://handbookofrobotics.org/view-chapter/20/videodetails/246
VIDEO 247 RDP experimental results
available from http://handbookofrobotics.org/view-chapter/20/videodetails/247
VIDEO 248 Stenting deployment system
available from http://handbookofrobotics.org/view-chapter/20/videodetails/248
VIDEO 249 Bimanual dissection
available from http://handbookofrobotics.org/view-chapter/20/videodetails/249
VIDEO 250 First concentric tube robot teleoperation
available from http://handbookofrobotics.org/view-chapter/20/videodetails/250
VIDEO 251 Shoe decoration using concentric tube robot
available from http://handbookofrobotics.org/view-chapter/20/videodetails/251
VIDEO 252 Concentric tube robot at TEDMED 2010
available from http://handbookofrobotics.org/view-chapter/20/videodetails/252
VIDEO 253 Aiko obstacle-aided locomotion
available from http://handbookofrobotics.org/view-chapter/20/videodetails/253
VIDEO 254 Aiko sidewinding
available from http://handbookofrobotics.org/view-chapter/20/videodetails/254
VIDEO 255 Anna Konda – Motion
available from http://handbookofrobotics.org/view-chapter/20/videodetails/255

References

20.1 B.Y.S. Hirose, H. Yamada: Snake-like robots, IEEE Robot. Autom. Mag. **16**(1), 88–98 (2009)
20.2 S. Hirose: *Biologically Inspired Robots: Snake-Like Locomotors and Manipulators* (Oxford Univ. Press, New York 1993)
20.3 Y. Tanaka, M. Arai, S. Hirose, T. Shingo: Development of *Souryu-V* with mono-tread-crawlers and elastic-rods joint, IEEE Int. Workshop Saf. Secur. Rescue Robot. (2006)
20.4 Hirose Fukushima Lab: *Souryu Robot* http://www-robot.mes.titech.ac.jp/robot/snake/soryu/soryu.html
20.5 H. Ohno, S. Hirose: Design of slim slime robot and its gait of locomotion, Proc. IEEE/RSJ Int. Conf. Intel Robot. Syst. (2001) pp. 707–715
20.6 H. Yamada, M. Mori, K. Takita, S. Ogami, S. Hirose: Development of amphibious snake-like robot ACM-R5, 36th Int. Symp. Robot. (2005)
20.7 G.S. Chirikjian, J.W. Burdick: Design and experiments with a 30 DOF robot, Proc. IEEE ICRA (1992) pp. 113–119
20.8 J.W. Burdick: Robots that crawl, walk, and slither, Eng. Sci. **55**(4), 2–13 (1992)
20.9 G.S. Chirikjian: Inverse kinematics of binary manipulators using a continuum model, J. Intel. Robot. Syst. **19**, 5–22 (1997)
20.10 J. Bluck: Squaring off with a robotic serpent. http://astrobiology.nasa.gov/articles/2001/2/21/squaring-off-with-a-robotic-serpent/

Avidbots Ex1016
Page 533 of 2154

20.11 University of Pennsylvania: Modlab, http://modlabupenn.org/multimedia/
20.12 M. Yim: New locomotion gaits, Proc. IEEE ICRA (1994) pp. 2508–2514
20.13 M. Yim, D. Duff: Modular robots, IEEE Spectr. **39**(2), 30–34 (2002)
20.14 M. Park, M. Yim: Distributed control and communication fault tolerance for the CKBot, IEEE Int. Conf. Reconfig. Mech. Robot. (2009) pp. 682–688
20.15 M. Yim, B. Shirmohammadi, J. Sastra, M. Park, M. Dugan, C.J. Taylor: Towards robotic self-reassembly after explosion, Proc. IEEE/RSJ Int. Conf. Intel. Robot. Syst. (2007) pp. 2767–2772
20.16 S. Revzen, M. Bhoite, A. Macasieb, M. Yim: Structure synthesis on-the-fly in a modular robot, Int. Conf. Intel. Robot. Syst. (2011) pp. 4797–4802
20.17 C. Wright, A. Johnson, A. Peck, Z. McCord, A. Naaktgeboren, P. Gianfortoni, M. Gonzalez-Rivero, R.L. Hatton, H. Choset: Design of a modular snake robot, Int. Conf. Intel. Robot. Syst. (2007) pp. 2609–2614
20.18 A. Johnson, C. Wright, M. Tesch, K. Lipkin, H. Choset: *A Novel Architecture for Modular Snake Robots, Tech. Report CMU-RI-TR-11-29* (Carnegie Mellon Univ., Pittsburgh 2011)
20.19 K. Lipkin, I. Brown, A. Peck, H. Choset, J. Rembisz, P. Gianfortoni, A. Naaktgeboren: Differentiable and piecewise differentiable gaits for snake robots, Int. Conf. Intel. Robot. Syst. (2007) pp. 1864–1869
20.20 Biorobotics Laboratory, Carnegie Mellon University http://biorobotics.org (for videos of Choset's robots)
20.21 P. Liljeback, O. Stavdahl, A. Beitnes: SnakeFighter – Development of a water hydraulic fire fighting snake robot, 9th Int. Conf. Control Autom. Robot. Vision (2006) pp. 1–6
20.22 P. Liljeback, K.Y. Pettersen, O. Stavdahl, J.T. Gravdahl: Experimental investigation of obstacle-aided locomotion with a snake robot, IEEE Trans. Robot. **99**, 1–8 (2011)
20.23 P. Liljeback, S. Fjerdingen, K.Y. Pettersen, Ø. Stavdahl: A snake robot joint mechanism with a contact force measurement system, Proc. IEEE ICRA (2009) pp. 3815–3820
20.24 J. Borenstein, M. Hansen, A. Borrell: The OmniTread OT-4 serpentine robot–design and performance, J. Field Robot. **24**(7), 601–621 (2007)
20.25 J.C. McKenna, D.J. Anhalt, F.M. Bronson, H.B. Brown, M. Schwerin, E. Shammas, H. Choset: Toroidal skin drive for snake robot locomotion, Proc. IEEE ICRA (2008) pp. 1150–1155
20.26 H. Date, Y. Takita: An electricity-free snake-like propulsion mechanism driven and controlled by fluids, Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst. (2009) pp. 3637–3642
20.27 T. Sato, T. Kano, A. Ishiguro: A decentralized control scheme for an effective coordination of phasic and tonic control in a snake-like robot, Bioinspir. Biomim. **7**(1), 016005 (2012)
20.28 T. Kamegawa, T. Yarnasaki, H. Igarashi, F. Matsuno: Development of the snake-like rescue robot *kohga*, Proc. IEEE Int. Conf. Robot. Autom. (2004) pp. 5081–5086
20.29 M. Hara, S. Satomura, H. Fukushima, T. Kamegawa, H. Igarashi, F. Matsuno: Control of a snake-like robot using the screw drive mechanism, Proc. IEEE Int. Conf. Robot. Autom. (2007) pp. 3883–3888
20.30 G. Robinson, J.B.C. Davies: Continuum robots – A state of the art, Proc. IEEE Int. Conf. Robot. Autom., Detroit (1999) pp. 2849–2854
20.31 G.S. Chirikjian: Theory and applications of hyper-redundant robotic mechanisms (Department of Applied Mechanics, California Institute of Technology, Pasadena 1992)
20.32 V.C. Anderson, R.C. Horn: Tensor arm manipulator design, Mech. Eng. **89**(8), 54–65 (1967)
20.33 B. Roth, J. Rastegar, V. Scheinman: On the design of computer controlled manipulators, 1st CISM-IFTMM Symp. Theory Pract. Robot. Manip. (1973) pp. 93–113
20.34 W.M. Kier, K.K. Smith: Tongues, tentacles and trunks: The biomechanics of movement in muscular-hydrostats, Zool. J. Linneaan Soc. **83**, 307–324 (1985)
20.35 F. Martin, C. Niemitz: How do African elephants (Loxodonta Africana) optimize goal-directed trunk movements?, Jahresvers. Dt. Zool. Ges. Dt. Ges. Parasitol. **96**, 159 (2003)
20.36 Y. Yekutieli, R. Sagiv-Zohar, B. Hochner, T. Flash: Dynamics model of the octopus arm. II. Control of reaching movements, J. Neurophysiol. **94**, 1459–1468 (2005)
20.37 R. Cieslak, A. Morecki: Elephant trunk type elastic manipulator – A tool for bulk and liquid type materials transportation, Robotica **17**, 11–16 (1999)
20.38 M.W. Hannan, I.D. Walker: Analysis and experiments with an elephant's trunk robot, Adv. Robot. **15**(8), 847–858 (2001)
20.39 H. Tsukagoshi, A. Kitagawa, M. Segawa: Active hose: An artificial elephant's nose with maneuverability for rescue operation, Proc. IEEE Int. Conf. Robot. Autom., Seoul (2001) pp. 2454–2459
20.40 E. Guglielmino, N. Tsagarakis, D.G. Caldwell: An octopus-anatomy inspired robotics arm, Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst., Taipei (2010) pp. 3091–3096
20.41 W. McMahan, B.A. Jones, I.D. Walker, V. Chitrakaran, A. Seshadri, D. Dawson: Robotic manipulators inspired by cephalopod limbs, Proc. CDEN Des. Conf., Montreal (2004) pp. 1–10
20.42 I.D. Walker, D. Dawson, T. Flash, F. Grasso, R. Hanlon, B. Hochner, W.M. Kier, C. Pagano, C.D. Rahn, Q. Zhang: Continuum robot arms inspired by cephalopods, Proc. 7th SPIE Conf. Unmanned Ground Veh. Technol., Orlando (2005) pp. 303–314
20.43 D. Trivedi, C.D. Rahn, W.M. Kier, I.D. Walker: Soft robotics: Biological inspiration, state of the art, and future research, Appl. Bionics Biomech. **5**(2), 99–117 (2008)
20.44 R.J. Webster III, B.A. Jones: Design and kinematic modeling of constant curvature continuum

Avidbots Ex1016
Page 534 of 2154

robots: A review, Int. J. Robot. Res. **29**(13), 1661–1683 (2010)
20.45 D.B. Camarillo, C.F. Milne, C.R. Carlson, M.R. Zinn, J.K. Salisbury: Mechanics modeling of tendon-driven continuum manipulators, IEEE Trans. Robot. **24**(6), 1262–1273 (2008)
20.46 L. Cowan: *Analysis and experiments for tendril-type robots*, M.S. Thesis (Clemson University, Clemson 2008)
20.47 M.W. Hannan, I.D. Walker: Kinematics and the implementation of an elephant's trunk manipulator and other continuum style robots, J. Robot. Syst. **20**(2), 45–63 (2003)
20.48 G. Immega: Tentacle-like manipulators with adjustable tension lines, U.S. Patent 5317952A (1992)
20.49 J.S. Mehling, M.A. Diftler, M. Chu, M. Valvo: A minimally invasive tendril robot for in-space inspection, Proc. Conf. BioRobotics (2006) pp. 690–695
20.50 A. Grzesiak, R. Becker, A. Verl: The bionic handling assistant – A success story of additive manufacturing, Assem. Autom. **31**(4), 329–333 (2011)
20.51 D.M. Lane, J.B.C. Davies, G. Robinson, D.J. O'Brien, J. Sneddon, E. Seaton, A. Elfstrom: The AMADEUS dextrous subsea hand: Design, modeling, and sensor processing, IEEE J. Ocean. Eng. **24**(1), 96–111 (1999)
20.52 M.B. Pritts, C.D. Rahn: Design of an artificial muscle continuum robot, Proc. IEEE Int. Conf. Robot. Autom., New Orleans (2004) pp. 4742–4746
20.53 K. Suzumori, S. Iikura, H. Tanaka: Development of flexible microactuator and its applications to robotic mechanisms, Proc. IEEE Int. Conf. Robot. Autom., Sacramento (1991) pp. 1622–1627
20.54 D.B. Camarillo, C.R. Carlson, J.K. Salisbury: Task-space control of continuum manipulators with coupled tendon drive, 11th Int. Symp. Exp. Robot. (2009) pp. 271–280
20.55 G. Immega, K. Antonelli: The KSI tentacle manipulator, Proc. IEEE Int. Conf. Robot. Autom., Nagoya (1995) pp. 3149–3154
20.56 I. Gravagne, C. Rahn, I.D. Walker: Large deflection dynamics and control for planar continuum robots, IEEE/ASME Trans. Mechatron. **8**(2), 299–307 (2003)
20.57 B.A. Jones, W. McMahan, I.D. Walker: Design and analysis of a novel pneumatic manipulator, Proc. 3rd IFAC Symp. Mechatron. Syst., Sydney (2004) pp. 745–750
20.58 W. McMahan, B.A. Jones, I.D. Walker: Design and implementation of a multi-section continuum robot: Air-Octor, Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst., Edmonton (2005) pp. 3345–3352
20.59 J. Lock, G. Laing, M. Mahvash, P.E. Dupont: Quasistatic modeling of concentric tube robots with external loads, Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst., Taipei (2010) pp. 2325–2332
20.60 L.G. Torres, R. Alterovitz: Motion planning for concentric tube robots using mechanics-based models, Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst., San Francisco (2011) pp. 5153–5159
20.61 R.J. Webster III, J.M. Romano, N.J. Cowan: Kinematics and calibration of active cannulas, Proc. IEEE Int. Conf. Robot. Autom., Pasadena (2008) pp. 3888–3895
20.62 R.S. Penning, J. Jung, J.A. Borgstadt, N.J. Ferrier, M.R. Zinn: Towards closed loop control of a continuum robotic manipulator for medical applications, Proc. IEEE Int. Conf. Robot. Autom., Shanghai (2011) pp. 4822–4827
20.63 B. Bardou, P. Zanne, F. Nageotte, M. de Mathelin: Control of multiple sections flexible endoscopic system, Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst., Taipei (2010) pp. 2345–2350
20.64 G. Chen, P.M. Tu, T.R. Herve, C. Prelle: Design and modeling of a micro-robotic manipulator for colonoscopy, 5th Int. Workshop Res. Educ. Mechatron., Annecy (2005) pp. 109–114
20.65 K. Xu, J. Zhao, J. Geiger, A.J. Shih, M. Zheng: Design of an endoscopic stitching device for surgical obesity treatment using a N.O.T.E.S. approach, Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst., San Francisco (2011) pp. 961–966
20.66 H.-S. Yoon, S.M. Oh, J.H. Jeong, S.H. Lee, K. Tae, K.-C. Koh, B.J. Yi: Active bending robot endoscope system for navigation through sinus area, Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst., San Francisco (2011) pp. 967–972
20.67 L.A. Lyons, R.J. Webster III, R. Alterovitz: Planning active cannula configurations through tubular anatomy, Proc. IEEE Int. Conf. Robot. Autom., Anchorage (2010) pp. 2082–2087
20.68 S. Wakimoto, K. Suzumori: Fabrication and basic experiments of pneumatic multi-chamber rubber tube actuator for assisting colonoscope insertion, Proc. IEEE Int. Conf. Robot. Autom., Anchorage (2010) pp. 3260–3265
20.69 N. Simaan, R. Taylor, P. Flint: A dexterous system for laryngeal surgery, Proc. IEEE Int. Conf. Robot. Autom., New Orleans (2004) pp. 351–357
20.70 H. Watanabe, K. Kanou, Y. Kobayashi, M.G. Fujie: Development of a 'Steerable Drill' for ACL reconstruction to create the arbitrary trajectory of a bone tunnel, Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst., San Francisco (2011) pp. 955–960
20.71 K. Xu, R.E. Goldman, J. Ding, P.K. Allen, D.L. Fowler, N. Simaan: System design of an insertable robotic effector platform for single port access (SPA) surgery, Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst., St. Louis (2009) pp. 5546–5552
20.72 J. Ding, K. Xu, R. Goldman, P. Allen, D. Fowler, N. Simaan: Design, simulation and evaluation of kinematic alternatives for insertable robotic effectors platforms in single port access surgery, Proc. IEEE Int. Conf. Robot. Autom., Anchorage (2010) pp. 1053–1058
20.73 P. Sears, P.E. Dupont: Inverse kinematics of concentric tube steerable needles, Proc. IEEE Int. Conf. Robot. Autom. (2007) pp. 1887–1892
20.74 P. Sears, P.E. Dupont: A steerable needle technology using curved concentric tubes, Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst. (2006) pp. 2850–2856
20.75 R. Buckingham: Snake arm robots, Ind. Robot An Int. J. **29**(3), 242–245 (2002)

Avidbots Ex1016
Page 535 of 2154

20.76 M. Mahvash, P.E. Dupont: Stiffness control of a continuum manipulator in contact with a soft environment, Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst., Taipei (2010) pp. 863–870

20.77 J. Jung, R.S. Penning, N.J. Ferrier, M.R. Zinn: A modeling approach for continuum robotic manipulators: Effects of nonlinear internal device friction, Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst., San Francisco (2011) pp. 5139–5146

20.78 D.C. Rucker, B.A. Jones, R.J. Webster III: A model for concentric tube continuum robots under applied wrenches, Proc. IEEE Int. Conf. Robot. Autom., Anchorage (2010) pp. 1047–1052

20.79 T. Aoki, A. Ochiai, S. Hirose: Study on slime robot: Development of the mobile robot prototype model using bridle bellows, Proc. IEEE Int. Conf. Robot. Autom., New Orleans (2004) pp. 2808–2813

20.80 H. Ohno, S. Hirose: Design of slim slime robot and its gait of locomotion, Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst., Maui (2001) pp. 707–715

20.81 B.A. Jones, M. Csencsits, W. McMahan, V. Chitrakaran, M. Grissom, M. Pritts, C.D. Rahn, I.D. Walker: Grasping, manipulation, and exploration tasks with the OctArm continuum manipulator, Proc. Int. Conf. Robot. Autom., Orlando (2006)

20.82 W. McMahan, M. Pritts, V. Chitrakaran, D. Dienno, M. Grissom, B. Jones, M. Csencsits, C.D. Rahn, D. Dawson, I.D. Walker: Field trials and testing of *OCTARM* continuum robots, Proc. IEEE Int. Conf. Robot. Autom. (2006) pp. 2336–2341

20.83 J.K. Salisbury: Whole arm manipulation, 4th Symp. Robot. Res. (1987)

20.84 I.D. Walker: Continuum robot appendages for traversal of uneven terrain in in-situ exploration, IEEE Aerosp. Conf. (2011) pp. 1–8

20.85 R. Kang, A. Kazakidi, E. Guglielmino, D.T. Branson, D.P. Tsakiris, J.A. Ekaterinaris, D.G. Caldwell: Dynamic model of a hyper-redundant, octopus-like manipulator for underwater applications, Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst., San Francisco (2011) pp. 4054–4059

20.86 I.S. Godage, D.T. Branson, E. Guglielmino, G.A. Medrano-Cerda, D.G. Caldwell: Shape function-based kinematics and dynamics for variable-length continuum robotic arms, Proc. IEEE Int. Conf. Robot. Autom., Shanghai (2011) pp. 452–457

20.87 R. Alterovitz, A. Lim, K. Goldberg, G.S. Chirikjian, A.M. Okamura: Steering flexible needles under Markov motion uncertainty, Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst. (2005) pp. 120–125

20.88 N.J. Cowan, K. Goldberg, G.S. Chirikjian, G. Fichtinger, R. Alterovitz, K.B. Reed, V. Kallem, W. Park, S. Misra, A.M. Okamura: Robotic needle steering: Design, modeling, planning, and image guidance. In: *Surgical Robotics – Systems, Applications, and Visions*, ed. by J, Rosen, B. Hannaford, R. Satava (Springer New York **pp**, 557–582 (2011)

20.89 W. Park, Y. Wang, G.S. Chirikjian: The path-of-probability algorithm for steering and feedback control of flexible needles, Int. J. Robot. Res. **29**(7), 813–830 (2010)

20.90 W. Park, J.S. Kim, Y. Zhou, N.J. Cowan, A.M. Okamura, G.S. Chirikjian: Diffusion-based motion planning for a nonholonomic flexible needle model, Proc. IEEE Int. Conf. Robot. Autom., Barcelona (2005)

20.91 R.J. Webster III, J.-S. Kim, N.J. Cowan, G.S. Chirikjian, A.M. Okamura: Nonholonomic modeling of needle steering, Int. J. Robot. Res. **25**(5–6), 509–525 (2006)

20.92 C. Rucker, R.J. Webster III, G.S. Chirikjian, N.J. Cowan: Equilibrium conformations of concentric-tube continuum robots, Int. J. Robot. Res. **29**(10), 1263–1280 (2010)

20.93 P.E. Dupont, J. Lock, B. Itkowitz, E. Butler: Design and control of concentric-tube robots, IEEE Trans. Robot. **26**(2), 209–225 (2010)

20.94 J. Lock, G. Laing, M. Mahvash, P.E. Dupont: Quasistatic modeling of concentric tube robots with external loads, 2010 IEEE/RSJ Int. Conf. Intell. Robot. Syst. (2010) pp. 2325–2332

20.95 P.E. Dupont, J. Lock, E. Butler: Torsional kinematic model for concentric tube robots, Proc. IEEE Int. Conf. Robot. Autom. (2009) pp. 3851–3858

20.96 C. Bedell, J. Lock, A. Gosline, P.E. Dupont: Design optimization of concentric tube robots based on task and anatomical constraints, Proc. IEEE Int. Conf. Robot. Autom. (2011) pp. 398–403

20.97 M. Mahvash, P.E. Dupont: Stiffness control of surgical continuum manipulators, IEEE Trans. Robot. **27**(2), 334–345 (2011)

20.98 A. Degani, H. Choset, A. Wolf, M.A. Zenati: Highly articulated robotic probe for minimally invasive surgery, Proc. IEEE Int. Conf. Robot. Autom. (2006) pp. 4167–4172

20.99 A. Degani, H. Choset, A. Wolf, T. Ota, M.A. Zenati: Percutaneous intrapericardial interventions using a highly articulated robotic probe, In The First IEEE/RAS-EMBS Int. Conf. Biomed. Robot. Biomech. (2006) pp. 7–12

20.100 R.J. Murphy, M.S. Moses, M.D. Kutzer, G.S. Chirikjian, M. Armand: Constrained workspace generation for snake-like manipulators with applications to minimally invasive surgery, Proc. IEEE Int. Conf. Robot. Autom. (2013) pp. 5341–5347

20.101 M.S. Moses, M.D. Kutzer, H. Ma, M. Armand: A continuum manipulator made of interlocking fibers, Proc. IEEE Int. Conf. Robot. Autom. (2013) pp. 4008–4015

20.102 S.M. Segreti, M.D.M. Kutzer, R.J. Murphy, M. Armand: Cable length estimation for a compliant surgical manipulator, Proc. IEEE Int. Conf. Robot. Autom. (2012) pp. 701–708

20.103 R.J. Murphy, M.D. Kutzer, S.M. Segreti, B.C. Lucas, M. Armand: Design and kinematic characterization of a surgical manipulator with a focus on treating osteolysis, Robotica **32**(6), 835–850 (2014)

Avidbots Ex1016
Page 536 of 2154

20.104 D. Trivedi, A. Lotfi, C.D. Rahn: Geometrically exact dynamics for soft robotics manipulators, Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst., San Diego (2007) pp. 1497–1502

20.105 G.S. Chirikjian, J.W. Burdick: A modal approach to hyper-redundant manipulator kinematics, IEEE Trans. Robot. Autom. **10**(3), 343–354 (1994)

20.106 I.A. Gravagne, I.D. Walker: Manipulability, force, and compliance analysis for planar continuum manipulators, IEEE Trans. Robot. Autom. **18**(3), 263–273 (2002)

20.107 B.A. Jones, I.D. Walker: Kinematics for multisection continuum robots, IEEE Trans. Robot. **22**(1), 43–55 (2006)

20.108 H. Mochiyama, T. Suzuki: Dynamic modeling of a hyper-flexible manipulator, Proc. 41st SICE Annu. Conf., Osaka (2002) pp. 1505–1510

20.109 H. Mochiyama, T. Suzuki: Kinematics and dynamics of a cable-like hyper-flexible manipulator, Proc. IEEE Intl. Conf. Robot. Autom., Taipei (2003) pp. 3672–3677

20.110 G.S. Chirikjian: Variational analysis of snakelike robots. In: *Redundancy in Robot Manipulators and Multi-Robot Systems*, Lecture Notes in Electrical Engineering, Vol. 57, ed. by D. Milutinovic, J. Rosen (Springer, New York 2013) pp. 77–91

20.111 G.S. Chirikjian: Hyper-redundant manipulator dynamics: A continuum approximation, Adv. Robot. **9**(3), 217–243 (1995)

20.112 I.S. Godage, E. Guglielmino, D.T. Branson, G.A. Medrano-Cerda, D.G. Caldwell: Novel modal approach for kinematics of multisection continuum arms, Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst., San Francisco (2011) pp. 1093–1098

20.113 G.S. Chirikjian, J.W. Burdick: Kinematics of hyper-redundant locomotion with applications to grasping, Proc. IEEE Int. Conf. Robot. Autom. (1991) pp. 720–725

20.114 M. Csencsits, B.A. Jones, W. McMahan: User interfaces for continuum robot arms, Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst., Edmonton (2005) pp. 3011–3018

20.115 E. Tatlicioglu, I.D. Walker, D.M. Dawson: Dynamic modeling for planar extensible continuum robot manipulators, Proc. IEEE Int. Conf. Robot. Autom., Rome (2007) pp. 1357–1362

20.116 W. Khalil, G. Gallot, O. Ibrahim, F. Boyer: Dynamic modeling of a 3-D serial eel-like robot, Proc. IEEE Int. Conf. Robot. Autom., Barcelona (2005) pp. 1282–1287

20.117 N. Giri, I.D. Walker: Three module lumped element model of a continuum arm section, Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst., San Francisco (2011) pp. 4060–4065

20.118 J. Li, J. Xiao: Determining 'grasping' configurations for a spatial continuum manipulator, Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst., San Francisco (2011) pp. 4207–4214

20.119 J. Xiao, R. Vatcha: Real-time adaptive motion planning for a continuum manipulator, Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst., Taipei (2010) pp. 5919–5926

20.120 M. Ivanescu, N. Bizdoaca, D. Pana: Dynamic control for a tentacle manipulator with SMA actuators, Proc. IEEE Int. Conf. Robot. Autom., Taipei (2003) pp. 2079–2084

20.121 M. Ivanescu, V. Stoian: A variable structure controller for a tentacle manipulator, Proc. IEEE Int. Conf. Robot. Autom., Nagoya (1995) pp. 3155–3160

20.122 A. Bajo, N. Simaan: Finding lost wrenches: Using continuum robots for contact detection and estimation of contact location, Proc. IEEE Int. Conf. Robot. Autom., Anchorage (2010) pp. 3666–3672

20.123 H. Mochiyama: Whole-arm impedance of a serial-chain manipulator, Proc. IEEE Int. Conf. Robot. Autom., Seoul (2001) pp. 2223–2228

20.124 D.C. Rucker, R.J. Webster III: Deflection-based force sensing for continuum robots: A probabilistic approach, Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst., San Francisco (2011) pp. 3764–3769

20.125 D. Braganza, D.M. Dawson, I.D. Walker, N. Nath: Neural network grasping controller for continuum robots, Proc. 45th IEEE Conf. Decis. Control, San Diego (2006)

20.126 A. Kapadia, I.D. Walker: Task space control of extensible continuum manipulators, Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst., San Francisco (2011) pp. 1087–1092

20.127 S. Ma: Analysis of creeping locomotion of a snake-like robot, Adv. Robot. **15**(2), 205–224 (2001)

20.128 K.Y. Pettersen, O. Stavdahl, J.T. Gravdahl: *Snake Robots: Modelling, Mechatronics, and Control* (Springer, London 2012)

20.129 A.A. Transeth, K.Y. Pettersen, P. Liljeback: A survey on snake robot modeling and locomotion, Robot. **27**(7), 999–1015 (2009)

20.130 R. Vaidyanathan, H.J. Chiel, R.D. Quinn: A hydrostatic robot for marine applications, Robot. Auton. Syst. **30**(1), 103–113 (2000)

20.131 C. Wright, A. Johnson, A. Peck, Z. McCord, A. Naaktgeboren, P. Gianfortoni, M. Gonzalez-Rivero, R. Hatton, H. Choset: Design of a modular snake robot, Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst. (2007) pp. 2609–2614

20.132 M. Tesch, K. Lipkin, I. Brown, R. Hatton, A. Peck, J. Rembisz, H. Choset: Parameterized and scripted gaits for modular snake robots, Adv. Robot. **23**(9), 1131–1158 (2009)

20.133 R.L. Hatton, H. Choset: Generating gaits for snake robots: Annealed chain fitting and keyframe wave extraction, Auton. Robot. **28**(3), 271–281 (2010)

20.134 J.C. McKenna, D.J. Anhalt, F.M. Bronson, H.B. Brown, M. Schwerin, E. Shammas, H. Choset: Toroidal skin drive for snake robot locomotion, Proc. IEEE Int. Conf. Robot. Autom. (2008) pp. 1150–1155

20.135 A. Wolf, H.B. Brown, R. Casciola, A. Costa, M. Schwerin, E. Shamas, H. Choset: A mobile hyper redundant mechanism for search and rescue tasks, Proc. 3rd IEEE/RSJ Int. Conf. Intell. Robot. Syst. (2003) pp. 2889–2895

20.136 C. Wright, A. Buchan, B. Brown, J. Geist, M. Schwerin, D. Rollinson, H. Choset: Design and archi-

Avidbots Ex1016
Page 537 of 2154

tecture of the unified modular snake robot, Proc. IEEE Int. Conf. Robot. Autom. (2012) pp. 4347–4354
20.137 J.M. Snyder, J.F. Wilson: Dynamics of the elastica with end mass and follower loading, J. Appl. Mech. **57**, 203 (1990)
20.138 J.F. Wilson, D. Li, Z. Chen, R.T. George Jr.: Flexible robot manipulators and grippers: Relatives of elephant trunks and squid tentacles. In: *Robots and Biological Systems: Towards a New Bionics?*, (Springer, Berlin, Heidelberg 1993) pp. 475–494
20.139 F. Naccarato, P.C. Hughes: Inverse kinematics of variable-geometry truss manipulators, J. Robot. Syst. **8**(2), 249–266 (1991)
20.140 P.C. Hughes, W.G. Sincarsin, K.A. Carroll: Trussarm – A variable-geometry-truss manipulator, J. Intell. Mater. Syst. Struct. **2**(2), 148–160 (1991)
20.141 R.J. Salerno, C.F. Reinholtz, S.G. Dhande, R. Hall: Kinematics of long-chain variable geometry truss manipulators: An overview of solution techniques, Proc. 2nd Int. Workshop Adv. Robot Kinemat. (1990)
20.142 A.S. Boxerbaum, K.M. Shaw, H.J. Chiel, R.D. Quinn: Continuous wave peristaltic motion in a robot, Int. J. Robot. Res. **31**(3), 302–318 (2012)
20.143 G.S. Chirikjian: Framed curves and knotted DNA, Biochem. Soc. Trans. **41**, 635–638 (2013)
20.144 I. Ebert-Uphoff, G.S. Chirikjian: Discretely actuated manipulator workspace generation by closed-form convolution, ASME J. Mech. Des. **120**(2), 245–251 (1998)
20.145 I. Ebert-Uphoff, G.S. Chirikjian: Inverse kinematics of discretely actuated hyper-redundant manipulators using workspace densities, Proc. IEEE Int. Conf. Robot. Autom. (1996) pp. 139–145
20.146 I. Ebert-Uphoff: On the development of discretely-actuated hybrid-serial-parallel manipulators (Department of Mechanical Engineering, Johns Hopkins University, Baltimore 1997)
20.147 Y. Wang, G.S. Chirikjian: Workspace generation of hyper-redundant manipulators as a diffusion process on SE(N), IEEE Trans. Robot. Autom. **20**(3), 399–408 (2004)
20.148 Y. Zhou, G.S. Chirikjian: Conformational statistics of semi-flexible macromolecular chains with internal joints, Macromolecules **39**(5), 1950–1960 (2006)

Avidbots Ex1016
Page 538 of 2154

Multimedia Contents

21. Actuators for Soft Robotics

Alin Albu-Schäffer, Antonio Bicchi

Although we do not know as yet how robots of the future will look like exactly, most of us are sure that they will not resemble the heavy, bulky, rigid machines dangerously moving around in old-fashioned industrial automation. There is a growing consensus, in the research community as well as in expectations from the public, that robots of the next generation will be physically compliant and adaptable machines, closely interacting with humans and moving safely, smoothly and efficiently – in other terms, robots will be *soft*.

This chapter discusses the design, modeling and control of actuators for the new generation of soft robots, which can replace conventional actuators in applications where rigidity is not the first and foremost concern in performance. The chapter focuses on the technology, modeling, and control of *lumped parameters* of soft robotics, that is, systems of discrete, interconnected, and compliant elements. *Distributed parameters*, snake-like and continuum soft robotics, are presented in Chap. 20, while Chap. 23 discusses in detail the biomimetic motivations that are often behind soft robotics.

21.1 **Background** 500
21.1.1 Motivations 500
21.1.2 Safety 500
21.1.3 Resilience 501
21.1.4 Performance and Energy Efficiency 501
21.1.5 The *Third Way* of Robotic Control and λ 501

21.2 **Soft Robot Design** 502
21.2.1 Natural Muscles 502
21.2.2 Breaking Down Artificial Muscle Systems 502
21.2.3 Sources for Soft Actuators 503
21.2.4 Pneumatic and Hydraulic 503
21.2.5 Electromagnetic and Electrostatic 503
21.2.6 Piezoelectric 504
21.2.7 Electro- and Magneto-Strictive 504
21.2.8 Shape Memory Alloys and Polymers 505
21.2.9 Polymeric Actuators 505
21.2.10 Carbon Nanotubes 505
21.2.11 Energy Storage 505
21.2.12 Energy Dissipation and Dampers 505
21.2.13 Artificial Muscles Arrangements 506
21.2.14 Specifications and Performance 507

21.3 **Modeling Actuators for Soft Robotics** 508
21.3.1 The Mass–Spring–Damper Paradigm 508
21.3.2 Nonlinear Mechanical Systems 508

21.4 **Modeling Soft Robots** 511

21.5 **Stiffness Estimation** 513
21.5.1 Observers for Linear Impedance 514
21.5.2 Variable Stiffness Observers 514
21.5.3 Parametric Estimation 514
21.5.4 Nonparametric Estimation 515

21.6 **Cartesian Stiffness Control** 515
21.6.1 Cartesian Impedance Control 515
21.6.2 Independent Position and Stiffness Control 516

21.7 **Periodic Motion Control** 518
21.7.1 Periodic Motion Tracking and Stiffness Adaptation 518
21.7.2 Controlled Oscillatory Dynamics 519
21.7.3 Exciting Intrinsic Oscillatory Dynamics 520
21.7.4 Model-Based Approach 520
21.7.5 Model-Free Approach 520

Avidbots Ex1016
Page 539 of 2154

21.8 **Optimal Control of Soft Robots** 521
21.8.1 OC Theory 521
21.8.2 SEA Optimization 521
21.8.3 Safe Brachistochrone OC Problem 522
21.8.4 Max Speed OC Problem 522
21.8.5 VSA Optimization 523
21.9 **Conclusions and Open Problems** 524
Video-References 525
References 526

The chapter starts by analyzing the main motivations that are behind the development of actuators for soft robotics: these are mainly safety of operation, especially in interaction with humans; resilience of the robot itself in interaction with harsh, unpredictable environments, and performance enhancement in dynamic tasks.

We then briefly survey the different principles and technologies that can be used to implement actuators for soft robotics. Using natural muscles as a term of comparison, we consider different elements constituting an actuator, such as the prime movers, energy storage and dissipation elements, and their different possible arrangements.

We then move to consider a class of actuators for soft robotics that includes many of the recent developments in this field, that is, the class of variable impedance actuators (VIA) and the subclass of variable stiffness actuators (VSAs), and consider how a mathematical model of these systems can be provided in general but precise terms. With the objective of helping the reader navigate the space of the many existing, and many more conceivable, designs we consider these actuators from a general point of view, discussing how their performance envelope and specifications could be given in a unified, comprehensive way.

In the final part of the chapter, we consider the new problems and opportunities arising from the control of soft robots, including the need to estimate the variable physical compliance of the actuators, to plan its variations to adapt to tasks and environments, and to do real-time control of soft robots exploiting their physical properties.

21.1 Background

21.1.1 Motivations

The intrinsic properties of the biological muscles and their advantageous arrangement within the musculoskeletal system are a key aspect to the motion performance of many animal species. The flexibility, force-to-weight ratio, viscous damping, and fast reacting properties provide animals with an actuator having properties well suited for the tasks they are faced with day to day. Taking the human as an example, our muscles allow us to do very fine manipulation, precisely exerting forces and torques at the millinewton level, but also lift objects at a 10 : 1 force-to-weight ratio. Through the interplay of these physical properties with the neural sensory-motor control, motion is accomplished in a very energy-efficient, safe, and effective way.

Despite the diverse range of actuation techniques developed over the past years, the lack of an actuator which can reach the functional performance of the biological muscle and its neuro-mechanical control system still remains one of the most significant barriers for the development of machines, which can match the motion capabilities of natural systems. This observation has motivated many researchers in recent years to work toward developing and exploiting actuation technologies for a new generation of soft robots, which can co-exist and co-operate with people and reach or even surpass their performance.

21.1.2 Safety

One of the main motivations for revisiting traditional robotic design is the concern about their safety: it is to be expected that interacting with robot assistants or collaborators should not constitute (or even just be perceived as) a higher injury risk to humans than the interaction with another cautious human. Attention toward safe operation of robots near to, and even in contact with, humans is rather recent, although some of the pioneering contributions date back to the 1990s [21.1]. Robot-specific risk assessment methods have been studied in recent years [21.2, 3], and are discussed in Chap. 69.

The inherent danger to humans of conventional arms can be mitigated by retro-fitting their design by increasing their sensorization (using, e.g., proximity-sensitive skins), or by increasing the energy-absorbing properties of protective layers (adding enough soft and compliant coverings or placing airbags around the arm).

Avidbots Ex1016
Page 540 of 2154

Starting in the early 1990s, researchers have gradually abandoned the retro-fitting approach and started designing robots with interaction safety in mind from the beginning: notable examples are MIT's WAM (whole arm manipulator) [21.4] and DLR's LWR (light-weight robot) [21.5]. Arms in this class are primarily characterized by low inertia of their moving parts (links and motors) and backdrivability. Advanced sensing and control can realize a *soft* behavior via software.

Another approach to increase the safety level of robot arms interacting with humans is *soft robotics*, that is, to intentionally introduce mechanical compliance and damping in design. By this researchers tend to replace the sensor-based computation of a behavior and its error-prone realization using active actuator control, by its direct physical embodiment, as in the natural example. Having compliance and damping in the robot structure is by no means sufficient to ensure its safety, as it might indeed be even counterproductive for the elastic energy potentially stored. Just like a human arm, a soft robot arm will need intelligent control to make it behave softly as when caressing a baby, or strongly as when punching.

21.1.3 Resilience

Physical interaction of a robot with its environment can also be dangerous to the robot itself. Indeed, while fortunately the number of accidents involving robots and humans is very limited, the number of times a robot is damaged because of impacts or force overexertion is rather large, and clearly the more so the less structured is the task and/or the environment. A typical cause of breakdown for traditional robots is the failure of reduction gears (especially harmonic drives are quite sensitive to shock loads) and motor bearings, but large accelerations can easily damage sensors and electronics. Although seldom considered in the scientific literature, resilience to shocks is not only instrumental to achieve viable applications of robots in everyday life, but would also be very useful in industrial environments, substantially enlarging the scope of applicability of robot technology. Soft robotics technologies can provide solutions that are extremely effective in absorbing shocks and reducing accelerations: soft materials can be used as coverings or even as structural elements in robot limbs, but the main technological challenge remains with soft actuators and transmissions.

21.1.4 Performance and Energy Efficiency

A further reason to search for a new generation of actuators with controllable compliance is for their dynamic behavior, which can afford high-performance, natural-looking motion while being more energy-efficient than rigid robots.

Indeed, adaptable compliance can be used to adjust the natural dynamics of the robot in tune with those of its environment, so that the intrinsic physical behavior of the resulting system is close to the desired motion. In these circumstances, actuators would only have to inject and extract energy into and out of the system for small corrective actions, thus reducing energy consumption.

The idea of embodying desirable dynamics in the physical properties of soft robots is of particular importance, for example, in walking/running robots and prostheses: indeed, biomechanical studies reveal that compliance adaptation is important for human walking and running. Although most humanoid robots (such as HRP-4 [21.6, 7] and ASIMO [21.8, 9]) use rigid robotics technologies, soft actuators are more and more often adopted in humanoid robots: pneumatic muscles are used, for example, in [21.10] and [21.11], while several legged systems have appeared that use series-elastic actuators [21.12]. What is probably the first prototype of fully physically compliant humanoid has been recently presented in [21.13].

The fact that natural systems change the compliance of their muscular system depending on the gait and environmental conditions, and even during the different phases of the gait, seems to indicate the potential usefulness of variable impedance actuators (VIA) for locomotion – yet, because of the technical and theoretical challenges posed, only early prototypes of bipedal walking systems with variable stiffness actuation have been reported so far [21.14, 15], and no humanoid robot has been reported (to the best of our knowledge) making full use of VIAs.

Besides their role in reducing energy consumption in cyclic motions, soft robotics can provide definite performance advantages in many other tasks. One notable class are those tasks where performance is measured in terms of peak velocity reached at the end of an explosive-type motion, such as for example, in throwing, hitting, jumping, kicking, etc. Compliance in the robot structure indeed allows to store elastic energy in a build-up phase, and restitute it in the subsequent explosive phase. Furthermore, it permits us to have joints that, at least for a short time duration, move at a substantially higher velocity than the motor shaft [21.16, 17].

21.1.5 The *Third Way* of Robotic Control and λ

Novel actuators for soft robotics introduce one further interesting innovation for what concerns their control

Avidbots Ex1016
Page 541 of 2154

philosophy and architecture. Indeed, it can be roughly but fairly said that most existing robotic systems are controlled according to either one of two dominant architectures: position-based controller and torque-based controller. Position-based controllers, which prevail in traditional industrial robotics, are rather closed controller architectures built around an inner servomechanism loop, which basically provide the user with the ability to set joint position references, with very limited access to inner variables such as velocities, gains, or torques. Torque-controlled robots, on the other hand, provide a completely open control architecture, whereby the user can specify the motor currents or joint torques directly as a function of all accessible states. While the position-based control is still by far the most widely used architecture, because it requires less costly sensors, actuators and processors, and is more robust, serviceable, and fool-proof, researchers have always had a preference for the versatility and openness of the torque-based control. The introduction of relatively large physical compliance in soft actuators changes the picture considerably. This is particularly true in VIA systems, where the user is provided with the control of neither the joint positions nor torques directly. Rather, one can specify the neutral equilibrium position of the joint, and the stiffness (and damping) by which the joint is attracted toward it, when external or inertial loads displace it from such equilibrium. To illustrate the difference, consider building a joint for a fully dynamic task, such as jumping on one leg: while position- or torque-control architectures would hardly be able to do the job (unless using very high-performance components), two relatively slow, low-cost servomotor units connected through springs in agonist–antagonist arrangement can already provide a very effective solution.

It should be noted here that such VIA control model is very similar in philosophy to the λ version of the equilibrium-point hypothesis in human motor control [21.18–20]. According to the equilibrium point hypothesis, movements arise as a consequence of shifts in the equilibrium of the motor system. This equilibrium is dependent upon the interaction of control signals, reflexes, muscle properties and loads, but it is under the control of central arm movement [21.21]. In this sense, soft robotics also provide a very interesting arena for exchanging ideas and insights with the research community in motor neuroscience.

21.2 Soft Robot Design

In this section, we provide a concise presentation of the main elements and technologies that may contribute to the development of novel actuators for soft robotics.

21.2.1 Natural Muscles

For comparison and inspiration, it is natural to consider first the characteristics of natural muscles. A typical muscle consists of many thousands of *muscle fibers* working in parallel, organized in *motor units*, that is, one or several fibers with a motor neuron activating them. A single muscle fiber contains several *myofibrils*, made of *sarcomere* separated by *Z disks*. In a sarcomere, *thin (actin) and thick (myosin) filaments* are arranged in a hexagonal lattice, such that one thick filament interacts with six thin filaments. *Cross-bridges* linking actin and myosin pull toward sacromere center from both ends resulting in muscle shortening. Actin and myosin filaments slide against each other byrepeated attachment and detachment of cross-bridges, thus resulting in muscle contraction.

As a consequence of their structure, natural muscles have several mechanical properties which differ from conventional actuators. Muscles (and tendons) have spring-like properties, with stiffness and damping increasing with activation, and are capable of storing and releasing energy, but also of dissipating it. Muscles are equipped with sensors which can provide stretch (e.g., spindles) and force (e.g., Golgi tendon organs) information to the nervous system. Muscle tension increases nonlinearly with activation, and – for fixed activation – depends on both length and velocity. Muscles are most often arranged in antagonistic pairs (such as the bicep and tricep in the elbow) or groups (such as in the shoulder): this fact, together with the nonlinearity of their force–stretch relationship, is used to modulate the effective impedance at the joints.

21.2.2 Breaking Down Artificial Muscle Systems

Even a nonexhaustive but reasonably broad analysis of technologies potentially contributing to the development of novel actuators for soft robotics could not fit this handbook chapter. However, we will attempt here to provide a simple categorization of the subject matter, along whose lines a more complete survey could be organized. To this purpose, we will break down a soft robot system into a network of simpler elements, whose behavior is simpler and more easily described.

Like other systems, a soft actuator is comprised of *storage* elements that keep energy inside the same energetic domain, and of *transducers* that allow a flow of energy from one domain to another. Transducers may dissipate energy via heat production (energy *sinks*), or can instead inject energy inside the system (energy *sources*). Sources can, in turn, be divided into *flow* sources, *effort* sources, and *hybrid*, or *real* sources. A flow source determines the amount of flow (linear or angular velocity, heat flow, volume or mass flow, . . .) going through it, without caring for the effort (force, torque, temperature, pressure, . . .) needed to keep such flow. The converse holds for an effort source. A hybrid source, instead, is a device that offers a generic constraint between the effort/flow pair.

Storage and transducer elements can be connected in different arrangements, which may result in completely different behaviors. VIA, an important class of actuators for soft robotics, have the characteristic that their overall behavior can be changed under active control between that of an effort and a flow source.

21.2.3 Sources for Soft Actuators

Technologies used as sources for soft actuators vary broadly in nature and scope. In this section, we limit ourselves to provide a list with comments on the range of applicability, while we refer the reader to specific literature for more detailed analysis: for instance, the author of [21.22] compares several active materials in a volumetrically normalized fashion, including stress (force-generating capability), strain (stroke length), speed of actuation, and power and energy metrics.

21.2.4 Pneumatic and Hydraulic

Pneumatic and hydraulic actuators [21.23, 24] include classical tie-rod or rod-less cylinders and rotary motors. Pneumatic actuators are usually low-impedance effort sources, while hydraulics tend to be high-impedance sources, but compliant behaviors can be actively controlled, or obtained through combination with other elements in the actuator system arrangement.

Of specific interest for soft robotics are the so-called *pneumatic artificial muscles* (PAMs), a name used to describe contractile linear-motion engines operated by gas pressure. The actuator's core element is a flexible membrane or shell attached at both ends to fittings along which mechanical power is transferred to a load. As the membrane is inflated, it bulges outward and expands radially, and contracts axially thus exerting a pulling force on its load. The force and motion thus generated by this type of actuator is typically unidirectional in tension (different from bellows, which work in compression). Pneumatic artificial muscles (PAM) became popular in robotics through the work of McKibben, who used braided sleeves around an elastic bladder as an orthotic actuator in the late 1950s [21.25], and have been widely used since then in robotics in various forms [21.26–32].

PAMs have an intrinsically nonlinear characteristic (force decreases with contraction) and, due to gas compressibility, they exhibit a natural low impedance. As they act one way, two PAM are needed to generate bidirectional motion in an antagonistic arrangement. By a pressure differential on the two muscles, a net joint torque is applied. On the other hand, due to nonlinearity of the stretch/force characteristic, an increase in the common-mode pressure varies the joint stiffness, while leaving torque unaffected. The above characteristics of PAMs are somehow similar to those natural muscles, although many important differences exist. From a practical viewpoint, the main drawbacks of pneumatic muscles are the encumbrance, noise, and slow response related to use of pressurized gas, and their relatively limited energy density.

21.2.5 Electromagnetic and Electrostatic

Electromagnetic motors use Lorentz-type forces acting on electric charges moving in a magnetic field. Examples are solenoids, that is, electromagnetically inductive coils wound around a movable steel or iron slug (the armature) [21.33], and rotating or linear induction motors [21.34]. Electromagnetic sources are typically effort sources, although often they have to be matched to the system impedance through gears, which may transform their behavior substantially. On the other hand, stepper motors, that is, brushless synchronous electric machines that divide a full rotation into a large number of steps, can be considered ideally as position sources. Stepper motors are mainly of three main types, that is, permanent magnet, hybrid synchronous, and variable reluctance steppers.

Electrostatic actuators are based on the attraction and repulsion of electric charges [21.35]. They typically require high-driving voltages at low currents (dual in a sense to electromagnetic motors). Linear electrostatic motors can be realized using comb-drive arrangements. While electrostatic drives at macroscopic scales are impractical due to large voltages, they have excellent potential at micro or nanoscale, where microelectromechanical systems (MEMSs) with moving, charged plates are far easier to fabricate than coils and iron cores.

Avidbots Ex1016
Page 543 of 2154

21.2.6 Piezoelectric

The *reverse piezoelectric effect*, that is the production of stress and/or strain in a piezoelectric material when an electric field is applied can also be used for actuating robots. Most used inorganic materials for piezoelectric transducres consist of ceramics with perovskite or tungsten-bronze structures, such as barium titanate ($BaTiO_3$), lead titanate ($PbTiO_3$) and lead zirconate titanate ($Pb[Zr_xTi_{1-x}]O_3$, $0 < x < 1$), commonly known as lead zirconate titanate (PZT) [21.36], while the direct effect of piezoelectric polymers such as polyvinylidene fluoride (PVDF) is mostly used for sensors.

Elementary piezoelectric actuators are best suited for microscale applications, where they can reach very high accuracy. Piezoelectric motors are sometimes used in resonant mode with rectification mechanisms that transform oscillations in net displacements (e.g., traveling-wave, inch-worm and stick-slip stepping motors). Multilayer (stacks) actuators, piling up several layers (usually thinner than 100 μm), are used to increase the actuator stroke, while retaining continuous and reversible operation. Stacked actuators are sometimes used within a mechanical amplifier structures, which can reach millimeter strokes (Fig. 21.1) Amplified piezoelectric stack actuators are commercially available from Dynamic Structures and Materials (Franklin, TN, USA) and Cédrat (Meylan, France). In essence, the compliant mechanism accomplishes a force–displacement tradeoff from input to output at the cost of storing energy within the compliant mechanism. This principle has been used in several mechanisms, such as in [21.37].

Ueda et al. [21.38] envision amplified piezoelectric stack actuators as modular units in a *cellular actuator*, drawing a parallel with muscles in that the overall actuation for a given joint is the sum of a number of discrete actionable units connected by compliant material. In the same way, muscles are composed of individual fibers grouped into motor units whose effects are coupled by elastic tendons. In order to function as modular, muscle-cell like units, the compliant mechanism in the amplified piezoelectric stacks need to have very aggressive force–displacement tradeoffs while fitting into a compact volumetric envelope. This is accomplished by using a hierarchical set of strain amplifiers, where the output of each stage in the amplification is the input to the subsequent stage. Each stage fits inside the subsequent stage; hence, this is referred to as a *nested* strain amplification mechanism. *Schultz* and *Ueda* [21.39, 40] present a framework to describe and specify the geometry of a nested configuration that results in a favorable tradeoff that meets design specifications. The work of *Schultz* and *Ueda* [21.41] draws yet another parallel with muscle, the idea that the central nervous system has authority over each motor unit in the muscle and activates each one in an on–off manner. This, in essence makes the control of muscles or muscle-like actuators a switching, rather than analog interface, and circumvents the problem of hysteresis inherent in positioning piezoelectric material with an analog voltage. This actuator type also incorporates this idea of redundancy; a lost motor unit (dead piezoelectric stack) does not result in total failure, but rather a loss of performance.

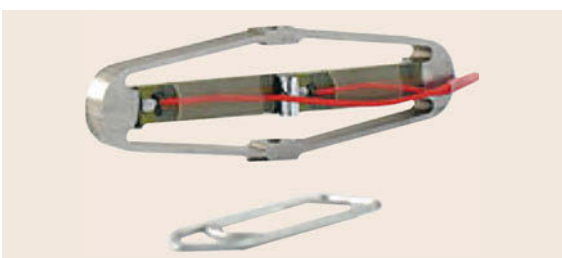


Fig. 21.1 Example of an amplified piezo actuator with multilayer stacks

21.2.7 Electro- and Magneto-Strictive

The electro- and magnetostrictive effects exhibited by some materials, that is, the tendency to change their shape under electric or magnetic fields, can also be exploited to build microscale actuators. Electrostriction differs from piezoelectricity because it is exhibited by all dielectric materials, and is proportional to the square of the polarization (instead of linearly). Reversal of the electric field does not reverse the direction of the deformation. Certain engineered ceramics, such as lead lanthanum zirconate titanate (PLZT) have high-electrostrictive constants and can produce strains of the order of 0.1% at field strengths of the order of 1 MV/m. Electrostrictive actuators are typically structured and used like piezoelectric actuators. Similarly, magnetostrictive materials deform under magnetic fields and in proportion to their magnetostrictive coefficient [21.42]. One of the strongest effects is exhibited by Terfenol-D, with strain of the order of 0.1% in a field of the order of 100 kA/m at room temperature.

The pronounced thermal expansion of some materials (e.g., doped single-crystal silicon or polysilicon) can be used to transform heat in mechanical energy and generate motion at micro scales. Temperature is typically controlled by Joules's effect, or through local heat injection, while mechanical amplification is often introduced with MEMS techniques (e.g., symmetric bent-beam or asymmetric bimorph structures).

Avidbots Ex1016
Page 544 of 2154

21.2.8 Shape Memory Alloys and Polymers

Macro-scale thermal actuators can otherwise be built using shape memory materials, which include alloys (shape memory alloys, SMA) and polymers (shape memory polymers SMP). Alloys like copper–zinc–aluminum–nickel, copper–aluminium–nickel, and nickel–titanium (NiTi) alloys [21.43] exhibit shape memory when transitioning from martensitic to austenitic phase (one-way effect) and, under particular conditions, also on the opposite phase change (two-way effect). SMA can exhibit up to 8% recoverable strain, and can be formed in virtually any shape. Energy inefficiency, slow response times, and large hysteresis are the main limitations. *Torres-Jara* et al. [21.44] have created an SMA muscle-like module in a convenient form factor by combining laser-cut SMA sheets with plastic hubs so that the hubs move away from one another when energized. These modules can be connected together in series or parallel to make an overall actuator with the desired properties.

SMPs are relatively new shape memory materials, which can be triggered by temperature but also by electric or magnetic field, light or solution [21.45, 46]. The shape change happens with transitions to glass phase, crystallization, or melting. While mechanical properties of SMPs are not yet comparable to SMAs, SMPs have an enormous potential for their versatility, as they range from stable to biodegradable, from soft to hard, and from elastic to rigid polymers.

21.2.9 Polymeric Actuators

Shape memory polymers represent only an example of viscoelastic materials which can be used for actuation. Very intense research has been pursued in recent years on actuators based on several different phenomena exhibited by polymeric materials [21.47]. Electroactive polymers (EAPs) are divided into ionic EAPs (based on diffusion of ions and solvents) and electronic EAPs (based electronic charging of the material). Ionic EAPs include polyelectrolyte gels [21.45, 46], ionic polymer metal composites (IPMC, such as Naftion by DuPont), conducting polymers (such as polypyrrole (PPy) and polyaniline (PANi)). Electronic EAPs include piezoelectric [21.48], electrostrictive, and dielectric polymers and elastomers (such as acrylic polymers and silicones [21.49]), and flexoelectric polymers, such as liquid crystal elastomers [21.50].

21.2.10 Carbon Nanotubes

Carbon nanotubes (CNTs) have a nonpolymeric macromolecular structure consisting of a graphite monoatomic sheet rolled to form tubes whose length can be about 1000 times their diameter. CNT actuators can be realized by using sheets of single- or multiwalled nanotubes as electrolyte-filled electrodes of a super capacitor [21.51], developing up to 0.75 MPa (cf. 0.3 MPa of the human muscle tissue) at low voltages, though still with limited stroke. A recent breakthrough in CNT actuators has been the use of highly ordered aerogel sheets, which allowed fabrication of actuators with giant strokes (ca. 180% actuation along the width) with fast response (5 ms delay) [21.52], which is slightly better than the human muscle. The artificial CNT muscle is stronger than steel in one direction, and more flexible than rubber in the other two directions [21.53]. When large strokes will be obtained at low voltages, the potential of CNT actuators for the design of artificial muscles is clearly enormous.

21.2.11 Energy Storage

Although energy storage elements in artificial muscles use a variety of materials and structures, due to the need of having mechanical energy readily available in dynamic phases, storage in the form of elastic energy prevails. This can be done by either pressurized gas systems (mainly in connection with hydraulic or pneumatic actuation systems), of by deformation of solid elements, that is, springs. Materials for storing a large amount of elastic energy in a small volume, and with limited losses, include special metal alloys (e.g., medium- and high-carbon steel, beryllium copper, and phosphor bronze). Composite materials, in the three main types available (polymer matrix composites (PMCs), metal matrix composites (MMCs) and ceramic matrix composites (CMCs)) offer a wider versatility in obtaining different stiffness, which depends not only on the constituent materials but also on the fiber design. Advanced needs of soft robotics applications motivate research in using sophisticated materials for energy storage, such as, for example, carbon nanotubes, elastomers, and superelastic materials.

Mechanical energy can also be stored in kinetic form through the use of inertial elements, similar to the kinetic energy recovery system (KERS) used in F1 racing cars. However, the mismatch with typical slow speeds of robotic joints introduces the need for gears. Conceptual design of CVT (continuous variable transmission)-based VIAs have been proposed, although research in this direction is still in its infancy.

21.2.12 Energy Dissipation and Dampers

Energy *sinks*, that is, dissipation systems and dampers, are another important element in the construction of

Avidbots Ex1016
Page 545 of 2154

a soft robot actuator. Indeed, underdamped behaviors are both nonnatural looking and potentially dangerous. Damping can be obtained through real-time control, or directly in hardware via electrical shunting or with mechanical devices. The latter have been extensively applied to a diverse range of cases, from the control of excessive structural response to transient environmental disturbances in civil structures, to automotive suspension systems, and more recently in robotics.

Dampers are often classified as either passive, semiactive or active systems, depending on the amount of external power required to perform its functionality. Dampers can be realized using different operating principles, which we briefly summarize here.

A friction damper (FD) is essentially composed of an actuator that applies a normal force on the output shaft, and generates friction as a consequence of relative motion ([21.54] for modeling). The control of the compression force between the frictional surfaces controls friction and can simulate damping: this has been effectively used in robotics to implement variable physical damping [21.55]. Drawbacks include the possibility for hysteresis and dead zone effects, due to static friction [21.56].

Electrorheological (ER) and magnetorheological (MR) dampers are based on liquids whose physical behavior depends on the application of electric or magnetic fields, respectively [21.56]. These fluids follow the Bingham models: after a yielding point, they behave as viscous liquids. The property that can be changed is the yield stress itself. The MR operating principle has been used to realize VDAs in vehicles [21.57] and also in robotics [21.58]. A more accurate model and a comparison between MR and FD can be found in [21.54], where it is pointed out that MR dampers, like the FDs, present high hysteresis.

Eddy current dampers (ECDs) are magnetic devices composed of a conductive material moving through a magnetic field. Eddy currents are induced and create a damping force that is proportional to the relative velocity between the material and the magnetic field. These devices can be realized with both permanent magnets and electromagnets. In both cases, there is the possibility of designing a device whose damping can be adjusted [21.59, 60]. In one case, the damping coefficient can be controlled by varying the intensity of the magnetic field, in the other case, by modifying the geometry of the conductor, or the gap between the conductor and the magnets (the effectiveness is shown in [21.60]). Electromagnetic ECDs have the advantage of not requiring mobile parts, but have the disadvantage of consuming power for maintaining a fixed damping value. Being fluid-free and contact-free, ECD dampers are clean and wear-free. However, they typically exert low damping torque at high velocity, thus requiring gearboxes for typical robotic applications.

Fluidic dampers are probably the most widely used in robotics and general machinery and can be divided into two main categories: turbulent flow dampers (high Reynolds number) which produce a damping force proportional to the square of the relative speed; and laminar flow dampers (low Reynolds number), where the damping force is proportional to relative speed. Simple dampers are largely used in the automotive industry using an orifice through which the viscous fluid flow is turbulent. Such device generates, at a given frequency, high damping for high amplitudes, but lower damping for lower amplitudes, and thus, has the drawback of presenting long lasting residual oscillations [21.56].

21.2.13 Artificial Muscles Arrangements

The basic technologies and elements reviewed above can be arranged in many different ways to achieve actuators for soft robotics. While a detailed taxonomy of arrangements is impossible here, the main distinction is between actuators that have a constant mechanical compliance characteristic, and those which can vary stiffness. In the first class are series elastic actuators (SEA), which have been developed since the 1990s for applications in humanoid locomotion and manipulation [21.61]. An SEA may include a conventional, rigid actuator connected to a compliant element, which is in turn connected on the other end to the moving link. The compliant element can exhibit a nonlinear compliance (load-deformation) characteristic, which however, is constant in time. SEA actuators, however, may use torque and position sensing at both ends of the elastic element and modify its natural mechanical impedance by virtue of active control of the motor. The impedance range to which an SEA can be effectively controlled is centered around the natural (passive) impedance of the elastic element and is limited in span by motor torque limits and overall control bandwidth.

As opposed, VSA can vary their natural impedance directly at the physical level, so that active control can be superimposed and ultimately achieve larger ranges of effective impedance than SEA. The idea of varying the mechanical impedance of actuation comes directly from natural musculoskeletal systems, which often exhibit this feature. VSA systems can change independently the equilibrium point of the moving body part, and the stiffness of the elastic force between the equilibrium point and the displaced body position (thus effectively implementing Feldman's *equilibrium point*, or λ-model for human movement control [21.62]). Accordingly, VSAs always use two prime movers. A rough distinction can be made between VSA ar-

Avidbots Ex1016
Page 546 of 2154

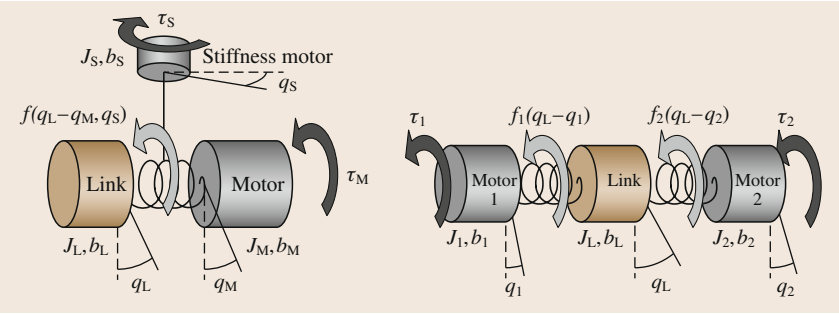


Fig. 21.2 Explicit stiffness variation and agonist–antagonist arrangements

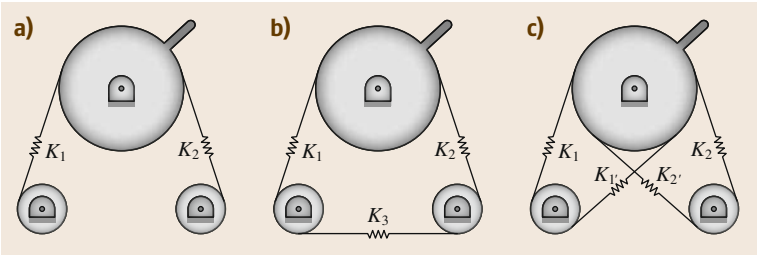


Fig.21.3a–c Three basic antagonist arrangements: **(a)** Simple, **(b)** cross-coupled and **(c)** bidirectional

rangements where the motors are explicitly dedicated to either equilibrium or stiffness control, and arrangements where the roles are mixed (Fig. 21.2). Explicit stiffness variation arrangements include, for example, the MIA (mechanical impedance adjuster) [21.63]), the AMASC (actuator with mechanically adjustable series compliance) [21.64], the MACCEPA (mechanically adjustable compliance and controllable equilibrium position actuator) [21.65]), and the variable stiffness joint (VS-Joint) [21.66]. Other VSA designs use instead agonist–antagonist (AA) arrangements, either in its simplest version, directly inspired to biological models, with two actuators pulling against each other [21.67, 68], or in other variants such as the cross-coupled AA [21.69] and bidirectional AA [21.70] arrangements (Fig. 21.3) It is a matter of simple calculations [21.71] to appreciate that, in an AA arrangement, the elastic elements must exhibit a nonlinear characteristic in order for the joint stiffness to be variable. If each tendon has a quadratic force–deformation relationship, then the overall joint behaves as a linear spring with variable spring constant K. A potential disadvantage of AA arrangements using unilateral (e.g., tendon) couplings between the prime movers and the driven link is that, to be able to exert a maximum torque τ at the link, two motors each capable of τ are needed. To alleviate this problem, the cross-coupled and bidirectional design introduce the possibility for an agonist actuator to also work in favour of the antagonist when required.

Recent work in soft robotics has introduced physically variable damping on the rotor dynamics in conjunction with series elastic or VSAs [21.72]. Physically variable damping based on the ECD effect has been demonstrated in haptic interfaces [21.73]. Some authors introduced physical dampers in robotic design, both in the so-called series damping actuator configuration [21.58], or as a variable friction damper connected to an SEA [21.55], and as a variable diaphragm laminar damper module connected to a variable stiffness actuator [21.74].

21.2.14 Specifications and Performance

The increasing number of available actuators for soft robotics and the widening scope of applications are rapidly creating a need for a simple language for application developers and actuator providers to talk together. Just like the perspective user of conventional hydraulic pistons or DC motors looks at spec sheets for a concise description of the performance guaranteed by the product, a soft robot designer will need to find salient facts about different artificial muscles presented in a clear and homogeneous way. The European project VIACTORS has dedicated a substantial effort to investigate what the crucial parameters of different designs are and has described a data sheet template that organizes these contents both from the designer and the user's points of view [21.75–77]. The interested reader is referred to these publications for details: it is interesting to note here that, besides the characteristics that are shared with conventional actuators (stall and peak motor torque, max velocity, power, electrical and mechanical interfaces, etc.), there are few data and curves

Avidbots Ex1016
Page 547 of 2154

that are specific of soft actuators: for instance, the range of stiffness variability, and the settling time needed to reach from minimum to maximum stiffness, or vice versa.

21.3 Modeling Actuators for Soft Robotics

To approach the analysis and control of novel actuators for soft robotics, it is necessary to describe their characteristics in terms of a mathematical model which accounts for their static and dynamic behaviors. We begin here by reviewing the familiar concept of mechanical impedance in linear time-invariant single degree of freedom (DOF) systems. Hence, in view of the fact that actuator softness control often introduces nonlinearity and time variance of the model, we proceed to generalize the concept. Finally, we move from consideration of a single actuator to modeling of multi-DOF, articulated soft robots.

21.3.1 The Mass–Spring–Damper Paradigm

The simplest example of mechanical impedance is a linear *spring*, that is, a sample of material which exhibits a proportional relation between applied force f and displacement at equilibrium y, $y = cf$. The proportionality constant c is the spring *compliance*. The inverse relation, that is, $f = ky$ introduces the spring *stiffness* $k = c^{-1}$. A spring can store an elastic energy V, whose first derivative with respect to the displacement y is the force

$$f = \frac{\partial V(y)}{y} ,$$

and the second derivative is the spring stiffness constant

$$k = \frac{\partial^2 V(y)}{\partial y^2} .$$

The concepts of compliance and stiffness are generalized from static (equilibrium) to dynamic cases by introducing the notion of *admittance* and *impedance*, respectively.

Consider a *damper* made of viscous material that opposes to displacements proportionally to how quick they are, that is, $f = b\dot{y}$, where b is the damping coefficient, and the inertial effects of accelerating an equivalent *mass* m of material under deformation, $f = m\ddot{y}$. In most materials, the three effects occur at the same time, and we have that the force/displacement relation is described by an ordinary differential equation. In the simple case of a single-DOF mass–spring–damper system, we have

$$f = m\ddot{y} + b\dot{y} + ky , \tag{21.1}$$

that is, a relation between two functions of time, $f(t)$ and $y(t)$. Such a differential equation describes a dynamical system. This can be regarded as a causal system if displacement (appearing with the highest-order derivative) is regarded as the effect (or output), and force as the cause (or input). Indeed, the knowledge of initial conditions on displacement, and the time course of force, determine all subsequent evolution of displacement.

Insofar as the dynamical system (21.1) is linear and time-invariant, its input–output behavior is conveniently studied via Laplace transforms from functions in the domain of time t to functions in the domain of the complex variable s. By denoting $F(s)$, $Y(s)$, $V_y(s)$ the transforms of $f(t)$, $y(t)$, and $v_y(t) = \dot{y}(t)$, respectively, one has immediately

$$\begin{aligned} F(s) &= (ms^2 + bs + k)Y(s) \\ &= (ms^2 + bs + k)s^{-1}V_y(s) . \end{aligned} \tag{21.2}$$

The operator $Z(s) := (ms^2 + bs + k)s^{-1}$ is called (mechanical) *impedance* of the spring–damper–mass system. The reciprocal operator of impedance is called *admittance* $A(s)$. Notice that the admittance operator is causal, while impedance is not.

21.3.2 Nonlinear Mechanical Systems

Real materials and systems deform under applied forces in more complicated ways than (21.1): in general, the dynamics between forces and deformations include nonlinearities, and can be of an order higher than two. How do we extend to these cases the concept of *impedance* (21.2), and specifically the notions of *mass, stiffness* and *damping*?

There is no easy clearcut solution to this problem. A straightforward approach is to find a linear, second-order model of type (21.1), which approximates well enough the real dynamics, and take its parameters to define those of the system. In this section we briefly review some of the implications of this approach.

Consider the relation between a force $f \in F$ (F here is the set of allowed forces) applied at some point in a material sample, and the displacement $y \in Y$ measured at equilibrium at the same point and along the same direction of the force (Y the set of allowed displacements). By ideally collecting all pairs $(f(t), y(t))$ corresponding in an infinite set of experiments (indexed

Avidbots Ex1016
Page 548 of 2154

by t for convenience), one can describe the relation by its graph $G \subset F \times Y$. Notice that this relation does not need to be a function: for hysteretic materials, for example, there are more admissible values of one variable corresponding to the same value of the other (Fig. 21.4).

Considering an analytic description of the graph G given by the locus $G(f, y) = 0$, at regular points of the graph a *force function* $f(y)$ is defined for values of y in a neighborhood of y_0. The stiffness associated with such a nonlinear force function can then be defined as

$$k(y) = \frac{\mathrm{d}f(y)}{\mathrm{d}y} = -\left(\frac{\partial G(f,y)}{\partial f}\right)^{-1} \frac{\partial G(f,y)}{\partial y} .$$

This can be generalized to a second-order, nonlinear dynamic setting by considering the relation between forces, displacements, and first- and second-order derivatives of displacements and its graph $G \subset F \times Y \times DY \times D^2Y$, comprised of 4-tuples $\mathrm{d}(t) := (f(t), y(t), \dot{y}(t), \ddot{y}(t))$ corresponding to an idealized, infinite set of experiments. (If hysteretic nonlinearities are considered, it is necessary to describe the relation in an extended space where hysteresis state variables are introduced [21.78].) If $G(f, y, \dot{y}, \ddot{y}) = 0$ is an analytical description of the graph, and d_0 is a regular point, then a *force function* $f(y, \dot{y}, \ddot{y})$ is defined in a neighborhood of d_0. More generally, when the system includes time-varying parameters $u(t)$ – such as, for example, a stiffening action in VSAs – one has $G(f, y, \dot{y}, \ddot{y}, u(t)) = 0$ and $f(y, \dot{y}, \ddot{y}, u)$. Following the linearization approach set out above, we can define admittance *along a given (nominal) trajectory* as the linear operator mapping small changes of the external force with respect to its nominal course, to changes in the resulting motion. To do this, consider the nonlinear ordinary differential equations (ODE) obtained by solving $G(f, y, \dot{y}, \ddot{y}, t) = 0$

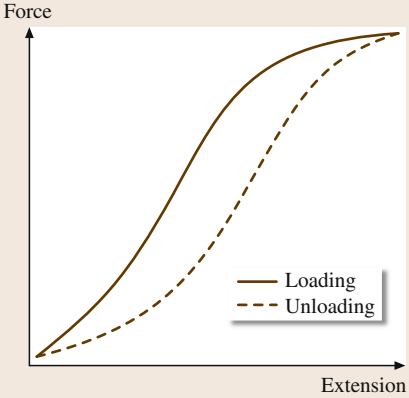


Fig. 21.4 Elastic hysteresis of a rubber-like material

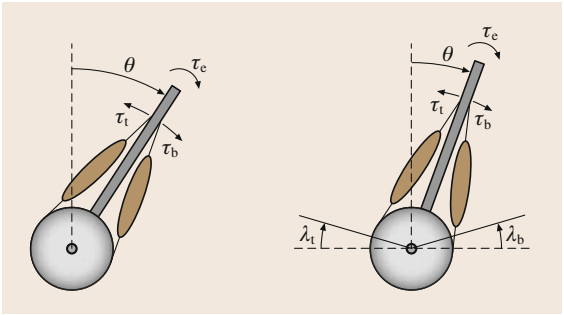


Fig. 21.5 A link subject to an external load and actuated by two antagonistic actuators. Nomenclature refers to actuator models used in (21.3) and (21.5), respectively

at a regular point with respect to $\ddot{y}$ as

$$\ddot{y} = g(y, \dot{y}, f, u) ,$$

and its state space form, with $x \in \mathbb{R}^2$, $x_1 = y$, $x_2 = \dot{y}$, that is,

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} x_2 \\ g(x_1, x_2, f, u) \end{pmatrix} .$$

For *given initial conditions* $x(0) = \bar{x}_0$ and a *given nominal course in time* of the force $\bar{f}(t)$, let $\bar{x}(t)$ be the nominal solution obtained. The first-order approximation of the dynamics of the perturbed motion $\tilde{x}(t) = x(t) - \bar{x}(t)$ corresponding to a change in force $\tilde{f}(t) = f(t) - \bar{f}(t)$ is the linear time-varying system

$$\dot{\tilde{x}} = \begin{pmatrix} 0 & 1 \\ -\kappa(t) & -\beta(t) \end{pmatrix} \tilde{x} + \begin{pmatrix} 0 \\ \mu(t) \end{pmatrix} \tilde{f} .$$

For the given system *and* the given motion, hence, we can define instantaneous mass, stiffness, and damping coefficients as

$$\begin{aligned} m(t) &= \mu^{-1}(t) ; \\ b(t) &= m(t)\beta(t) ; \\ k(t) &= m(t)\kappa(t) . \end{aligned}$$

Example 21.1

Consider the link in Fig. 21.5, actuated by two antagonistic actuators (with a role vaguely similar to that of the biceps and triceps muscles at the elbow), with a quadratic damping and subject to gravity and to an external torque load τ_e. The system dynamics are

$$I\ddot{\theta} + \beta\dot{\theta}|\dot{\theta}| - \tau_b + \tau_t - \mathrm{mgl}\sin\theta - \tau_e = 0 .$$

Avidbots Ex1016
Page 549 of 2154

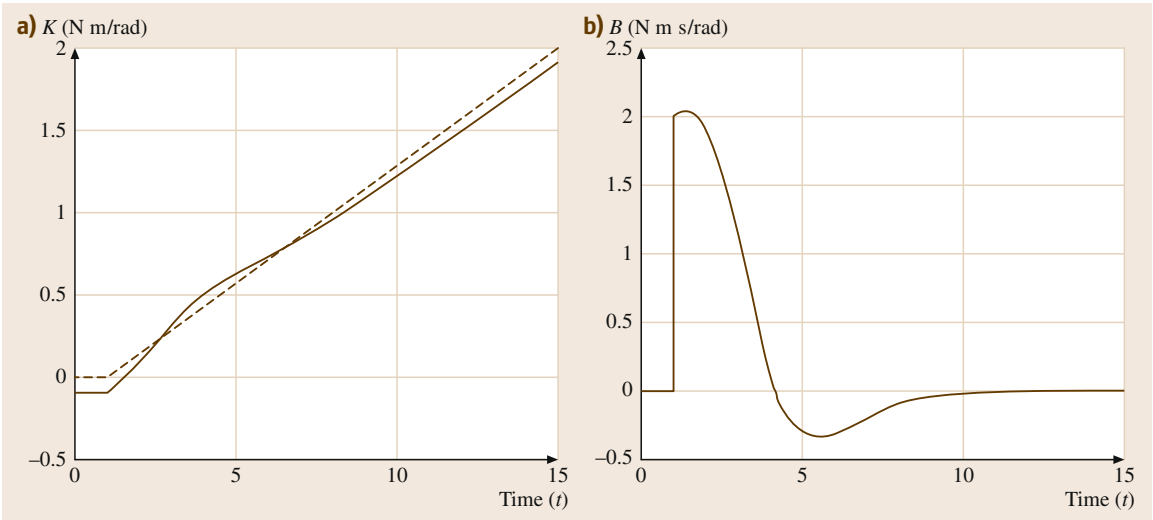


Fig.21.6a,b Generalized stiffness **(a)** (*dashed* is without gravity term) and generalized damping for the example with actuators as in (21.3), subject to a unit step in external torque at $t = 1$ s, and **(b)** with time-varying activation $u_b(t) = u_t(t)$ linearly increasing from 0 at $t = 1$ s to 1 at $t = 15$ s. Numerical values used in simulation: $I = 0.05\,\mathrm{Nm\,s^2}$, $mgl = 0.1\,\mathrm{Nm}$, $\beta = 1\,\mathrm{Nm\,s^2}$, $\alpha = 1\,\mathrm{Nm}$, $\tau_{max} = 2\,\mathrm{Nm}$

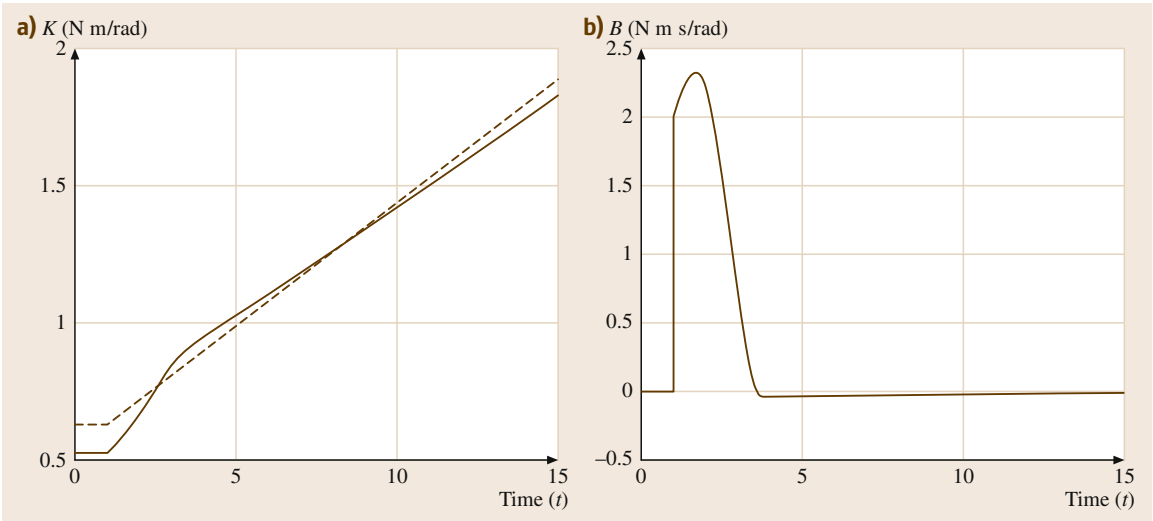


Fig.21.7a,b Generalized stiffness **(a)** (*dashed* is without gravity term) and generalized damping for the example with actuators as in (21.5), subject to a unit step in external torque at $t = 1$ s, and **(b)** with time-varying reference angle $u_b(t) = u_t(t)$ linearly decreasing from $\pi/3$ at $t = 1$ s to 0 at $t = 15$ s. Numerical values used in simulation as in Fig. 21.6, except for $\alpha = 0.3\,\mathrm{Nm}$

Assume first that the two actuators generate torques according to the model [21.67]

$$\tau_b = (\tau_{max} - \alpha\theta_b)\, u_b\,, \qquad \tau_t = (\tau_{max} - \alpha\theta_t)\, u_t\,, \tag{21.3}$$

where $\theta_b = (\pi/2 + \theta)$, $\theta_t = (\pi/2 - \theta)$, τ_{max} is the maximum isometric torque, u_b, u_t are the normalized contraction parameters ($0 \le u \le 1$, $u = u_b$, u_t), and α is a constant assumed to be equal for the two actuators. We easily obtain $m = I$ for the generalized mass, $b(\dot\theta) = 2\beta|\dot\theta|$ for the generalized damping, and

$$k(\theta, u) = \alpha(u_b + u_t) - \mathrm{mgl}\cos(\theta) \tag{21.4}$$

for the generalized stiffness. In the latter expression, the

Avidbots Ex1016
Page 550 of 2154

role of a gravity-induced term and that of a co-activation stiffness term are apparent.

If a different actuator model is adopted, namely [21.79]

$$\tau_b = -\alpha(\theta_b - u_b)^2 ,$$
$$\tau_t = -\alpha(\theta_t - u_t)^2 , \tag{21.5}$$

where u_b, u_t are now interpreted as the rest lengths of the actuators, one has

$$k(\theta) = 2\alpha(\pi - \lambda_b - \lambda_t) - \mathrm{mgl}\cos(\theta) .$$

The values of stiffness and damping for the two examples above, corresponding to time-varying values of the control parameters, are reported in Figs. 21.6 and 21.7, respectively. ■

So far, we have considered only impedance for second-order dynamical systems – basically modeling actuators with a single moving element that dominates the system inertia. In practice, one may encounter higher order systems because of either (i) more complex actuator systems, or (ii) the connection of actuators and links in an articulated robot structure. The latter case will be discussed in Sect. 21.4.

Examples of actuators with higher order dynamics are the arrangements in Fig. 21.3: if the agonist and antagonist prime movers as well as the driven link have nonnegligible inertia, the dynamics are at least of sixth order. If the system is linear, admittance can still be defined as the sixth-order transfer function from link torque to link displacement: however, there is no natural way of associating an equivalent single mass, spring or damper coefficient. To recover these notions, an approach could be to apply some form of model reduction of the higher order dynamics to second order [21.80]. However, model reduction is all but an obvious procedure, as the approximate heavily depends on how the similarity of response between dynamics is measured. With nonlinear systems, the problem is clearly even more complicated. Overall, the topic is definitely worth further investigation, which should take into account what are the ultimate practical goals of defining impedance parameters.

21.4 Modeling Soft Robots

In this section, a brief introduction of dynamic models for soft robots is given, starting with a quite general lumped-parameter formulation and successively introducing several simplifying assumptions.

A soft robot with lumped elasticities can be imagined as a set of directly actuated rigid bodies with configuration $\boldsymbol{\theta} \in \mathbb{R}^m$, connected to the indirectly actuated rigid bodies (with configuration vector $\boldsymbol{q} \in \mathbb{R}^n$) through viscoelastic forces, as exemplary sketched in Fig. 21.8. The entire configuration space of the system is denoted by $\boldsymbol{x} = (\boldsymbol{\theta}, \boldsymbol{q})$, $\boldsymbol{x} \in \mathbb{R}^{(n+m)}$. A quite general abstraction of a compliant robot, which can be used for the generic design of controllers, is given by

$$\mathbf{M}(\boldsymbol{x})\ddot{\boldsymbol{x}} + \mathbf{c}(\boldsymbol{x}, \dot{\boldsymbol{x}}) + \left(\frac{\partial V(\boldsymbol{x})}{\partial \boldsymbol{x}}\right)^{\mathrm{T}} + \boldsymbol{d}(\boldsymbol{x}, \dot{\boldsymbol{x}}) = \begin{pmatrix} \boldsymbol{\tau_m} \\ \boldsymbol{\tau}_{\mathrm{ext}} \end{pmatrix}, \tag{21.6}$$

with $\mathbf{M}(\boldsymbol{x})$ being the inertia matrix, $\boldsymbol{c}(\boldsymbol{x}, \dot{\boldsymbol{x}})$ the Coriolis and centrifugal vector, V the potential energy of the elastic element and of the gravity forces, $\boldsymbol{\tau}_m \in \mathbb{R}^m$ the actuator generalized forces acting as control inputs, and $\boldsymbol{\tau}_{\mathrm{ext}} \in \mathbb{R}^n$ the external torques acting on the robot as a disturbance. The most relevant property of this structure is its underactuation, meaning that the system has less control inputs ($\boldsymbol{m}$) than its configuration space dimension ($\boldsymbol{n} + \boldsymbol{m}$). However, in contrast to other purely inertially coupled underactuated systems (for example multiple pendulums such as the Acrobot [21.81].), for the considered robots $V(\boldsymbol{x})$ is [21.82] positive definite, implying that a unique equilibrium point exists for each external torque with actuators in a fixed configu-

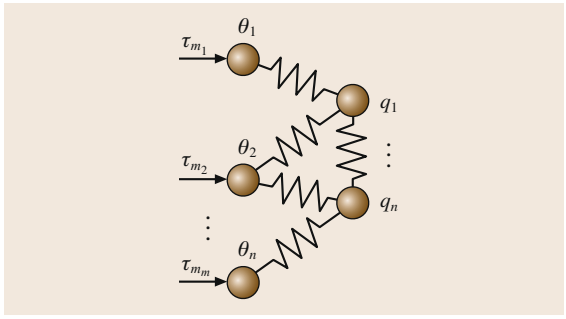


Fig. 21.8 A soft robot can be imagined as a set of directly actuated rigid bodies with configuration $\boldsymbol{\theta}$, connected to the indirectly actuated rigid bodies (with configuration coordinates $\boldsymbol{q}$ through elastic forces). Note that in general, inertial couplings, not depicted in the figure, might exist as well

Avidbots Ex1016
Page 551 of 2154

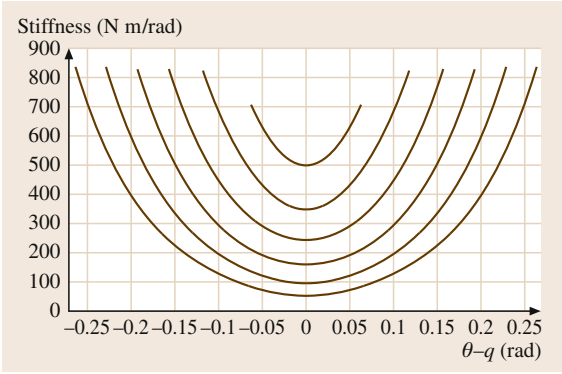


Fig. 21.9 Typical shapes of elastic torque characteristics, denoted by nonlinear deflection dependency and a family of linear curves, parameterized by $\boldsymbol{\sigma}$

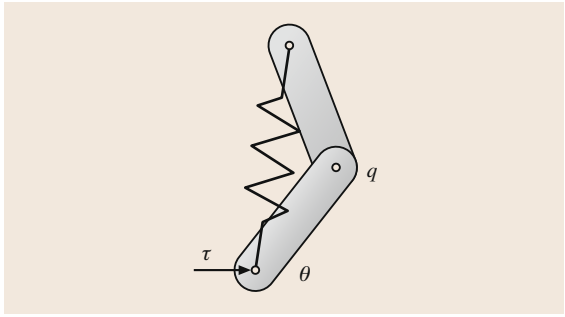


Fig. 21.10 In this example, the directly and the indirectly actuated states are both inertially and elastically coupled. This leads to a fully coupled inertia matrix in the dynamics equations

ration $\boldsymbol{\theta} = \boldsymbol{\theta}_0$ and that the linearization of the system around an equilibrium point $\{\boldsymbol{x} = \boldsymbol{x}_0, \dot{\boldsymbol{x}} = 0\}$ is controllable. Typically, $V(\boldsymbol{x}) = V_G(\boldsymbol{x}) + V_\tau(\boldsymbol{x})$, that is, the potential function is the sum of a gravity potential V_G and an elastic potential V_τ. The elastic potential function $V_\tau(\boldsymbol{x})$ is a convex function (Fig. 21.9), increasing strongly enough to compensate for the destabilizing effects of V_G, such that $V(\boldsymbol{x})$ is convex as well. Furthermore, the system contains in general a dissipative friction force $\boldsymbol{d}(\dot{\boldsymbol{x}}, \boldsymbol{x})$ with $\dot{\boldsymbol{x}}^T \boldsymbol{d}(\dot{\boldsymbol{x}}, \boldsymbol{x}) \leq 0$. Note that between the directly and indirectly actuated states, there can exist also inertial couplings, as depicted in Fig. 21.10. Therefore, $\mathbf{M}(\boldsymbol{x})$ is in general a fully coupled matrix.

While at first sight looking pretty general and simple, model (21.6) is actually quite complex due to the full, parallel inertia and stiffness coupling. This can be seen by rewriting it in the less compact form

$$\begin{pmatrix} \mathbf{M}(\boldsymbol{q}, \boldsymbol{\theta}) & \mathbf{Q}(\boldsymbol{q}, \boldsymbol{\theta}) \\ \mathbf{Q}^T(\boldsymbol{q}, \boldsymbol{\theta}) & \mathbf{B}(\boldsymbol{q}, \boldsymbol{\theta}) \end{pmatrix} \begin{pmatrix} \ddot{\boldsymbol{q}} \\ \ddot{\boldsymbol{\theta}} \end{pmatrix} + \boldsymbol{c}(\boldsymbol{q}, \boldsymbol{\theta}, \dot{\boldsymbol{q}}, \dot{\boldsymbol{\theta}}) + \boldsymbol{g}(\boldsymbol{q}, \boldsymbol{\theta})$$

$$+ \begin{pmatrix} \dfrac{\partial V(\boldsymbol{q}, \boldsymbol{\theta})^T}{\partial \boldsymbol{q}} \\ \dfrac{\partial V(\boldsymbol{q}, \boldsymbol{\theta})^T}{\partial \boldsymbol{\theta}} \end{pmatrix} + \boldsymbol{d}(\dot{\boldsymbol{q}}, \dot{\boldsymbol{\theta}}) = \begin{pmatrix} \boldsymbol{\tau}_{\text{ext}} \\ \boldsymbol{\tau}_m \end{pmatrix} . \qquad (21.7)$$

The above formulation emphasized the inertial coupling terms $\mathbf{Q}(\boldsymbol{q}, \boldsymbol{\theta})$ as well as possible dependencies of the inertial properties on both motor and link configuration.

Although (21.6) is the only formulation general enough to describe most soft robot designs with lumped elasticities encountered in the literature so far, it might pose substantial difficulties in controller design, that is, for the fact that it is not necessarily feedback-linearizable. (See, for example, the full dynamic model of flexible joint robots from [21.83, 84], which is not static state-linearizable.) Therefore, several simplifying assumptions are encountered in the literature, justified by particular robot design choices:

- *A1*: The system has no inertia coupling between the directly and the indirectly actuated stated, but only elastic couplings. This assumption includes the usual neglecting of motor gyroscopic effects for high-gear actuators [21.85] as well the neglection of small inertial couplings usually appearing in the VIA mechanisms [21.86] and imposes some restrictions related to mechanical design (Fig. 21.10 for a counter-example). The assumption leads to a block-diagonal structure of the mass matrix.
- *A2*: The gravity potential energy as well as the variation of the mass matrix is only due to the link-side variables $\boldsymbol{q}$. This implies that also the Coriolis and centrifugal torques are only depending on $\boldsymbol{q}$ and $\dot{\boldsymbol{q}}$.
- *A3*: The dynamics of the stiffness adjusting mechanism is negligible; the system dynamics is mainly described by the states $\boldsymbol{q}$ and a directly actuated state vector $\boldsymbol{\theta} \in \mathbb{R}^n$ of the same dimension as $\boldsymbol{q}$, corresponding to one principal motor per joint. The remaining $m-n$ motor coordinates of the stiffness adjusters are treated as parameters and are usually denoted by $\boldsymbol{\sigma}$. In this case, the elastic potential reduces to $V_\tau(\boldsymbol{q} - \boldsymbol{\theta}, \boldsymbol{\sigma})$. This is already a quite restricting assumption, applying mainly for SEA-type actuators with a dedicated stiffness adjuster, as typically used in the design of large joints of arms and legs. The model is practically valid only if stiffness is changed off-line or at low rate. Classical antagonistic actuators including two equally sized motors, as used, for example, in some robotic hand designs [21.87] do not fulfill this assumption.
- *A4*: A further particular choice of $V_\tau(\boldsymbol{q} - \boldsymbol{\theta}, \boldsymbol{\sigma})$ is that of a quadratic function in $\boldsymbol{q} - \boldsymbol{\theta}$ such that the

Avidbots Ex1016
Page 552 of 2154

joint torque is

$$\boldsymbol{\tau} = \frac{\partial V(\boldsymbol{\phi}, \boldsymbol{\sigma})^{\mathrm{T}}}{\partial \boldsymbol{\phi}} = \mathbf{K}(\boldsymbol{\sigma})\boldsymbol{\phi} \,, \tag{21.8}$$

with $\boldsymbol{\phi} = \boldsymbol{q} - \boldsymbol{\theta}$. This implies that the stiffness of the actuator depends only on the stiffness adjuster and not on the deflection of the springs [21.82]. This assumption introduces additional simplifications in controller design, applies however only to a restricted subclass of soft actuators.

The model, under assumptions *A1*, *A2* reduces to

$$\begin{pmatrix} \mathbf{M}(\boldsymbol{q}) & \mathbf{0} \\ \mathbf{0} & \mathbf{B} \end{pmatrix} \begin{pmatrix} \ddot{\boldsymbol{q}} \\ \ddot{\boldsymbol{\theta}} \end{pmatrix} + \begin{pmatrix} \boldsymbol{c}(\boldsymbol{q}, \dot{\boldsymbol{q}}) \\ \mathbf{0} \end{pmatrix} + \begin{pmatrix} \boldsymbol{g}(\boldsymbol{q}) \\ \mathbf{0} \end{pmatrix} + \begin{pmatrix} \dfrac{\partial V(\boldsymbol{q}, \boldsymbol{\theta})^{\mathrm{T}}}{\partial \boldsymbol{q}} \\ \dfrac{\partial V(\boldsymbol{q}, \boldsymbol{\theta})^{\mathrm{T}}}{\partial \boldsymbol{\theta}} \end{pmatrix} + \boldsymbol{d}(\dot{\boldsymbol{x}}) = \begin{pmatrix} \boldsymbol{\tau}_{\mathrm{ext}} \\ \boldsymbol{\tau}_m \end{pmatrix} \,. \tag{21.9}$$

Note that the model above does not make any further specification regarding the dimensions m and n, thus covering a broad range of robots, including as particular cases:

- The classical, fully antagonistic case with $m = 2n$.
- The quite popular $n + 1$ tendon design for robot fingers which leads to $m = n + 1$. More generally, $2n > m > n$ tendon designs are encountered.
- Synergy hands with $m < n$, where each synergistic actuator acts over springs on several links.
- Multi-articular muscles, as mostly existing in biological systems.

Under assumption *A3* the size of $\boldsymbol{\theta}$ reduces to the size of $\boldsymbol{q} (m = n)$ and the model becomes

$$\begin{pmatrix} \mathbf{M}(\boldsymbol{q}) & \mathbf{0} \\ \mathbf{0} & \mathbf{B} \end{pmatrix} \begin{pmatrix} \ddot{\boldsymbol{q}} \\ \ddot{\boldsymbol{\theta}} \end{pmatrix} + \begin{pmatrix} \boldsymbol{c}(\boldsymbol{q}, \dot{\boldsymbol{q}}) \\ \mathbf{0} \end{pmatrix} + \begin{pmatrix} \boldsymbol{g}(\boldsymbol{q}) \\ \mathbf{0} \end{pmatrix} + \begin{pmatrix} \boldsymbol{\tau} \\ -\boldsymbol{\tau} \end{pmatrix} + \boldsymbol{d}(\dot{\boldsymbol{x}}) = \begin{pmatrix} \boldsymbol{\tau}_{\mathrm{ext}} \\ \boldsymbol{\tau}_m \end{pmatrix} \,, \tag{21.10}$$

with $\boldsymbol{\tau}^{\mathrm{T}} = \partial V(\boldsymbol{\phi}, \boldsymbol{\sigma}) / \partial \boldsymbol{\phi}$. It is then quite close to the flexible joint models treated in detail in Chap. 11. The similarity becomes even stronger, when the torque is described by (21.8) under assumption *A4*. Note that the dimension of $\boldsymbol{\theta}$ in (21.10) is half of the dimension of the same variable in (21.9) since the dynamics of the stiffness adjuster is neglected.

21.5 Stiffness Estimation

Strictly speaking, impedance is a differential operator relating physical quantities (forces and displacements). Rather than a direct measurement, the process of characterizing impedance of a system is therefore a process of *dynamical system identification*.

In typical procedures for impedance measurement of material samples, such as, for example, organic tissue samples, the measured system is put into mechanical contact with another mechanical system, the instrument. The resulting compound dynamical system is then subject to a dynamical excitation of different nature, and data are collected. Experiments are typically conducted by touching the sample with a small, hard, vibrating probe. On the basis of measurements on the probe motion (displacement y, velocity $\dot{y}$ or acceleration $\ddot{y}$) and the applied force f, the mechanical impedance of the sample is inferred. A vast literature is available on the metrology of impedance of materials, and the reader is referred there for a detailed survey [21.88].

The problem of measuring impedance in articulated systems is obviously more complex than for material samples, but is of great relevance to several fields – including notably human movement science and neuroscience, who strive to understand how humans vary and control impedance in their movements. When the hand is slightly perturbed during arm movements, it tends to return to the undisturbed trajectory [21.20, 89], as if the hand would be connected to a spring along the planned trajectory. This spring-like property stems mainly from muscle elasticity and the stretch reflex, which produce – through the complex leverage of muscles at different limb configurations – a restoring force toward the undisturbed trajectory. Methods to characterize the mechanical impedance of human limbs have been the subject of extensive research [21.90–98]. Current protocols for identifying impedance in human motion typically recur to experiments in which perturbations are purposefully injected in the system, and their effects are measured. In artificial robotic systems, impedance parameters can also be calculated on the basis of a precise description of the model (wherever this is available), or obtained through accurate calibration procedures. In both natural and artificial systems, it would be of great utility to have a method which could identify impedance in real time, without per-

Avidbots Ex1016
Page 553 of 2154

turbing the normal execution of the task, and robustly to inaccurate modeling and to time-varying parameters altering calibration results. Of particular interest in robotics is the application of such methods to VIAs for soft robotics. As discussed, these can be implemented in a variety of ways and arrangements, but all designs share a fundamentally unavoidable nonlinear behavior, which makes stiffness measurement a tough problem.

According to the description of impedance as a differential operator, its characterization can be cast as an identification problem of a dynamical system. To date, the literature on this subject is not abundant, and it can be subdivided in two main approaches, that is, parametric and nonparametric identification. Parametric approaches postulate that a model of the actuator generating variable stiffness is known in its structure, but that some of its parameters are unknown. Nonparametric approaches do not use such an assumption, and are thus more general; however, as we will see, nonmodel-based observers have the disadvantage that they cannot *go to sleep* after reaching convergence, and require persistent excitation to stay tuned to variable impedance.

21.5.1 Observers for Linear Impedance

To introduce the problem, let us first consider the case of constant linear impedance identification in the elementary system (21.1)

$$f = m\ddot{y} + b\dot{y} + ky \,,$$

where we assume that m, b, k are unknown but constant, while y and f are available from position and force measurements at one of the ends of the compliant elements.

Before embarking in building an algorithm to estimate impedance, it is important to establish that the problem is well posed, that is, that a solution exists and is unique. To this purpose, consider an extended state vector

$$z = \begin{pmatrix} y & \dot{y} & -\frac{k}{m} & -\frac{b}{m} & \frac{1}{m} \end{pmatrix} \,,$$

and rewrite the dynamics (21.1) as a nonlinear dynamical system

$$\dot{z} = \begin{pmatrix} z_2 \\ z_1 z_3 + z_2 z_4 \\ 0 \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ z_5 \\ 0 \\ 0 \\ 0 \end{pmatrix} f, \qquad (21.11)$$

$$y = h(z) = z_1 \,.$$

The identification of the impedance parameters can thus be cast as a nonlinear observability problem, that is, from the knowledge of the input f and output y, estimate the initial state $z(0)$, and in particular its three last components which completely determine the linear impedance. The observability codistribution for this system is

$$\Omega(z) = \operatorname{span} \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ z_3 & z_4 & z_1 & z_2 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ z_3 z_4 & z_3 + z_4^2 & z_2 + z_1 z_4 & z_1 z_3 + 2 z_2 z_4 & 0 \\ 0 & 0 & 0 & z_5 & z_4 \end{pmatrix} .$$

It is easy to check that, for m, b, $k > 0$, the observability codistribution has codimension zero for all states, except if $z_1 = z_2 = 0$. Hence, whenever the system is not in stationary equilibrium, the three linear constant impedance parameters can, in principle, be reconstructed from position and force measurements.

To actually estimate the impedance in this case, different methods can be adopted. These include standard off-line identification techniques (which exploit the linear nature of the regressor for the unknown parameters), such as, for example, in [21.99], or on-line nonlinear state observers (e.g., extended Kalman filters) applied to system (21.11) [21.100].

21.5.2 Variable Stiffness Observers

Unfortunately, the generalization of the above straightforward approach to the case when impedance is nonlinear and/or time-varying is not trivial. To convince oneself, it is sufficient to consider the case that, instead of the linear spring term ky, a nonlinear and time-varying spring function $s(y, u(t))$ is introduced in (21.1), that is

$$f = m\ddot{y} + b\dot{y} + s(y, u) \,. \qquad (21.12)$$

In the $s(y, u(t))$ term, u indicates an exogenous input that commands changes in stiffness. For instance, disregarding gravity in Example 21.1 earlier, u would represent the co-activation term $u_\mathrm{b} + u_\mathrm{t}$ in (21.4).

21.5.3 Parametric Estimation

A first approach to identify stiffness can be based on the assumption that the arrangement of the variable stiffness actuator is known (Fig. 21.2), and that a parametric

description of the spring function is available, for example, in terms of a finite basis expansion. In this case, parametric stiffness observers have been proposed that work *on the motor side*, that is, considering the dynamics of each motor as

$$\tau = J\ddot{\theta} + b\dot{\theta} - f(\phi)\,, \tag{21.13}$$

where τ is the torque of the motor, J and b are the motor inertia and damping, respectively, and $\phi = \theta - q$ is the displacement angle with q being the link angle. These parameters and signals being known or measured, the problem is to estimate the stiffness $\sigma(\phi) = \frac{\partial f}{\partial \phi}$, assuming it can be approximated by the truncated Taylor expansion

$$\sigma_i(\phi_i) \approx \sum_{j=0}^{N} \alpha_j^i \frac{(\phi_i)^j}{j!}\,.$$

In [21.101], a parametric stiffness observer is presented to solve this problem. The idea is to first obtain an estimation of the spring force function $f(\phi)$ by combining a modified kinematic Kalman filter for the estimation of the first derivative of q and a first-order filter. Then $f(\phi)$ is approximated by a power series whose coefficients are estimated by a least-squares algorithm. Finally, σ is obtained by analytically differentiating the Taylor expansion. The method has been demonstrated to work well in simulation, and encouraging experimental results have been reported. The method is somewhat sensitive to the choice of several parameters, which require careful tuning. Subsequently, [21.102, 103] proposed algorithms based on differential algebra and modulating functions, respectively, that improved convergence and robustness avoiding taking any derivatives and are very easy to tune.

21.5.4 Nonparametric Estimation

When no information on the structure of f or on the variable $u(t)$ is available, the only possible approach is *on the link side*, that is trying to estimate the joint stiffness looking only at the link dynamics, without assuming any structure for the spring stiffness function. A nonparametric method was proposed in [21.104] to measure stiffness in a system such as (21.12), which can be easily derived from the idea of differentiating (21.12) once with respect to time

$$\dot{f} = m\dddot{y} + b\ddot{y} + \sigma\dot{y} + s_u\dot{u}\,,$$

where

$$s_u := \frac{\partial s(y,u)}{u}\,.$$

The method is based on the continuous update of an estimate $\hat{\sigma}$ of stiffness. Based on a best effort prediction for the link torque $\dot{f}$ as

$$\dot{\hat{f}} = m\dddot{y} + b\ddot{y} + \hat{\sigma}\dot{y}\,,$$

it was shown that, under mild assumptions, the update law

$$\dot{\hat{\sigma}} = \alpha(\dot{f} - \dot{\hat{f}})\mathrm{sgn}(y)\,, \tag{21.14}$$

with $\alpha > 0$ makes $\hat{\sigma}(t)$ converge to the true stiffness value $\sigma(t)$ within a uniformly ultimately bounded error. The error bound is affected by errors in obtaining filtered approximations of derivatives of the position $y(t)$ and of the applied force $f(t)$, and in estimating mass and damping coefficients m and b, respectively. Conversely, the error can be reduced by increasing the gain factor α. As opposed to parametric methods, no assumptions are made on the function $s(y,u)$, except that it is smooth and bounded in both arguments. The method has been experimentally demonstrated to work well, practically converging to the true stiffness value in all circumstances except when stiffness is varied without moving the link. In this case (where it is indeed theoretically impossible to observe stiffness *from the link side*), the method of [21.104] simply stops updating the stiffness estimate, to resume as soon as a sufficiently rich dynamic excitation is again applied to the link.

21.6 Cartesian Stiffness Control

21.6.1 Cartesian Impedance Control

The Cartesian impedance control paradigm is one of the most widely used manipulation control approaches in modern robotics to handle contact situations [21.90, 105, 106]. It combines the advantages of compliant manipulation with the ease and intuitiveness of a task description in Cartesian coordinates. Indeed, this control technique has been one of the motivations for the research of variable-impedance-actuated robots. Therefore, it is a natural aim to analyze and transfer Cartesian impedance control concepts to soft robots.

The goal addressed in this section is to realize a desired Cartesian impedance behavior at the tool center point (TCP) of a variable- impedance- actuated robot. Following a central paradigm of VIA, the mechanical

robot parameters need to be tuned such that the desired behavior is naturally achieved on a mechanism level to the largest extent possible.

With the model from (21.10), the VIA joints shall be adjusted to realize a Cartesian stiffness. The stiffness behavior is described by a constant stiffness matrix $\mathbf{K}_C = \partial \boldsymbol{f} / \partial \boldsymbol{x} \in \mathbb{R}^{m \times m}$ as the relation between the Cartesian wrench f and the Cartesian displacement x. The n passive and adjustable joint stiffness components provide the diagonal matrix $\mathbf{K}_J = \partial \boldsymbol{\tau} / \partial \boldsymbol{q} \in \mathbb{R}^{n \times n}$ with the joint torques τ and the joint positions q. Decoupled joint compliance is assumed. The mapping from the Cartesian stiffness space to the joint stiffness space is given by $\mathcal{T}: \mathbf{K}_J = \mathbf{T}(\mathbf{K}_C)$. This transformation can be written as

$$\begin{aligned}\mathbf{K}_J &= \frac{\partial \boldsymbol{\tau}}{\partial \boldsymbol{q}} = \frac{\partial (\mathbf{J}(\boldsymbol{q})^T \mathbf{K}_C \Delta \boldsymbol{x})}{\partial \boldsymbol{q}} \\ &= \mathbf{J}(\boldsymbol{q})^T \mathbf{K}_C \mathbf{J}(\boldsymbol{q}) - \frac{\partial \mathbf{J}(\boldsymbol{q})^T \mathbf{K}_C \Delta \boldsymbol{x}}{\partial \boldsymbol{q}} ; \end{aligned} \tag{21.15}$$

$\mathbf{J}(\boldsymbol{q}) = \frac{\partial f(\boldsymbol{q})}{\partial \boldsymbol{q}}$ is the manipulator Jacobian, where $\boldsymbol{f}(\boldsymbol{q})$ is the forward kinematics mapping. $\Delta \boldsymbol{x} = \boldsymbol{x}_d - \boldsymbol{x}$ is the Cartesian position error between the desired and the actual position. The first part of (21.15) reflects the stiffness around the equilibrium point. The second part of (21.15) is due to the change of the Jacobian [21.107].

The Cartesian stiffness at the equilibrium position ($\Delta \boldsymbol{x} = 0$), resulting from a specific joint stiffness, can be obtained by solving the inverse problem of (21.15), $\mathbf{K}_C = \mathcal{T}^{-1}(\mathbf{K}_J)$. Note that at the equilibrium position the second term in (21.15) vanishes. Using compliance matrices $\mathbf{C}_\bullet = \mathbf{K}_\bullet^{-1}$, it results from

$$\mathbf{C}_C = \mathbf{J}(\boldsymbol{q}) \mathbf{C}_J \mathbf{J}(\boldsymbol{q})^T \tag{21.16}$$

that

$$\mathbf{K}_C = (\mathbf{J}(\boldsymbol{q}) \mathbf{K}_J^{-1} \mathbf{J}(\boldsymbol{q})^T)^{-1} . \tag{21.17}$$

In [21.108], it was investigated that an arbitrary Cartesian stiffness matrix with three translational and three rotational stiffnesses ($\mathbf{K}_C \in \mathbb{R}^{6 \times 6}$) can hardly be reached by a 7-DOF robot, even exploiting the null space. Bounds are imposed by the technical realization, namely the diagonal joint stiffness matrix or mechanical VSA stiffness limits. Furthermore, the transformation of the joint stiffness into the Cartesian space is strongly determined by the robot kinematics and pose.

To overcome these restrictions and improve the stiffness tracking performance, one can use the passive stiffness in combination with an active impedance controller [21.109], which widely extends the achievable Cartesian stiffness range. Active control adapts the stiffness in a wider range, and the elastic elements are capable of absorbing impacts and increase the energy efficiency. By combining the two concepts, one can benefit from the individual advantages. The serial interconnection of the active stiffness $\mathbf{K}_{active}$ and the passive one $\mathbf{K}_{passive}$ results in an overall stiffness $\mathbf{K}_{res}$ (Fig. 21.11)

$$\mathbf{K}_{res}^{-1} = \mathbf{K}_{active}^{-1} + \mathbf{K}_{passive}^{-1} . \tag{21.18}$$

To compute the active and passive stiffness components, a two-step optimization algorithm can be used [21.109]. It achieves first a passive stiffness as close as possible to the desired one and secondly designs the active stiffness to minimize the residual.

21.6.2 Independent Position and Stiffness Control

Variable stiffness robots are well suited for manipulation tasks involving the interaction with an environment. Such tasks are often specified by a robot pose trajectory and an associated stiffness trajectory. As VSA robots allow us to vary joint position and stiffness in a mechanical way, a direct approach is to adjust the respective position or stiffness inputs. Depending upon the joint setup, these inputs are reflected by distinct actuators [21.66] or can be achieved by a simple coordinating transformation arising from the coupling of actuators [21.69]. The underlying idea is to embed and thereby achieve the desired robot behavior on a mechanical level. Additional control goals, especially important in the presence of disturbances, are the minimization of deflections and the suppression of vibrations.

The solutions to achieve independent control of link motion and stiffness can be classified in two categories:

- One class of controllers exploits the knowledge of system models.
  In [21.110] and [21.111], feedback linearization approaches are used to transform the robot dynamics

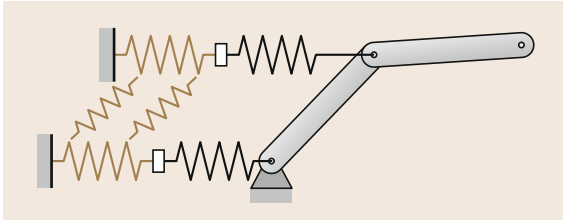


Fig. 21.11 Combination of active impedance controller (*brown* springs) with passive compliance (*black* springs) to achieve a desired Cartesian stiffness

Avidbots Ex1016
Page 556 of 2154

into an equivalent model of simpler form. A decoupled chain of integrators can be achieved, as long as system inversion is possible. The simple structure of the equivalent dynamics allows for simultaneous decoupling and accurate tracking of motion and stiffness reference profiles. However, the abstraction of the robot dynamics hinders the implementation of performance criteria.

Another model-based approach aims to achieve a reduced order model [21.112]. Therefore, separate dynamics are identified in the robot system, namely the arm, the positioning actuators, and the stiffness actuators. The independence of these dynamics is shown by a singular perturbation analysis. A cascaded control structure is based upon this analysis.

The abstraction that this class of controllers provides, allows for theoretical simplicity and design flexibility. On the other hand, robot performance and robustness in the presence of model uncertainty are not guaranteed. High-model accuracy and high derivatives of states are often required. Furthermore, to conform with the idea of embodiment additional effort is necessary, as the controllers often introduce a different robot behavior.

- The second class of controllers are energy-shaping-based controllers.

  One of the first controllers in this category was presented in [21.7]. The controller acts on motor position and uses a transformation to independently control the joint position and stiffness. The controller is validated on a one-DOF VSA joint.

  An extension and a generalization have been presented in [21.113]. The control design formulation is valid for a quite general form of underactuated Euler–Lagrange systems including variable impedance robots. Herein, the controller action can be interpreted as shaping of the potential and kinetic robot energy ensuring system passivity.

  A general task is described as to control k independent output variables given by $\boldsymbol{q} = \boldsymbol{h}(\boldsymbol{x})$ to desired constant values $\boldsymbol{q}_\mathrm{d} \in \mathbb{R}^k$. $\boldsymbol{x} \in \mathbb{R}^n$ is the vector of generalized coordinates, where $n = 2k$ is the usual case for VSA robots. Given the structure of VSA robots (21.10), a new variable $\bar{\boldsymbol{q}}$ can be found. This is a collocated (directly actuated) variable, which is statically equivalent to the noncollocated (indirectly actuated) $\boldsymbol{q}$. Using this collocated variable for a passive feedback ensures stability and can be interpreted as shaping the potential energy of the system.

  The variable $\bar{\boldsymbol{q}}$ is achieved by solving the static solution of the link side equation

$$\boldsymbol{\tau} + \boldsymbol{g}(\boldsymbol{q}) = 0 \tag{21.19}$$

  for q. Except for very simple cases, this equation has to be solved numerically. Due to the convex nature, this is a fast and numerically robust task in practice. Consequently, the controller

$$\boldsymbol{u} = \boldsymbol{g}(\bar{\boldsymbol{q}}) - \mathbf{J}_{\bar{q}}^{\mathrm{T}}(\boldsymbol{\theta})\mathbf{K}_\mathrm{p}(\bar{\boldsymbol{q}} - \boldsymbol{q}_\mathrm{d}) - \mathbf{K}_\mathrm{d}\dot{\boldsymbol{\theta}} \tag{21.20}$$

  stabilizes the desired position $\boldsymbol{q}_\mathrm{d}$; $\boldsymbol{g}(\bar{\boldsymbol{q}})$ is a feed forward term, compensating for the gravity. The use of $\bar{\boldsymbol{q}}$ enables arbitrarily low controller gains even for large displacements from the equilibrium. $\mathbf{J}_{\bar{q}}$ is a Jacobian mapping the collocated variable on the statical equivalent $\bar{\boldsymbol{q}}$. Global asymptotic stability based on LaSalle's invariance theorem can be shown.

  The approach can be extended in order to include also the feedback of torque and torque derivative, with the effect of reducing the apparent actuator inertia and friction. This allows improving the transient performance while remaining within the passivity framework. A torque feedback of the form

$$\boldsymbol{\tau}_m = \mathbf{B}\mathbf{B}_\theta^{-1}\boldsymbol{u} + (\mathbf{I} - \mathbf{B}\mathbf{B}_\theta^{-1})\boldsymbol{\tau}\ , \tag{21.21}$$

  with a new control input u leads to a new subsystem with scaled motor inertia

$$\mathbf{B}_\theta\ddot{\boldsymbol{\theta}} + \boldsymbol{\tau} = \boldsymbol{u}\ . \tag{21.22}$$

  The torque controller can therefore be interpreted as a scaling of the kinetic energy of the rotors in order to reduce the vibrations caused by the joint flexibility. The controlled system allows again a passive representation.

  The energy shaping approach provides excellent performance in the static case and for well damped systems. Some joints show low intrinsic damping to enable joint torque estimation and energy efficiency. In this case it is desirable to add additional damping via control or to include a physical variable damping element. Several damping control structures can be considered. A simple gain scheduling approach for a one-DOF system has been presented in [21.114]. Local linear sub problems are identified on which a linear quadratic regulator (LQR) state feedback controller is designed. The nonlinearity of the robot dynamics requires to adapt the controller poles dependent on the system state, which is especially hard for the multi-joint robots.

  A physically motivated state feedback control approach for multi-DOF VSA robots has been presented in [21.115] using an eigenvalue-based modal decomposition.

Avidbots Ex1016
Page 557 of 2154

21.7 Periodic Motion Control

Robotic tasks, such as walking, jumping, running, and some pick-and-place motions, are essentially periodic motions. During these tasks, the natural oscillatory dynamics of compliant actuators can be ideally exploited to increase performance and energy efficiency. It is well known that fast running animals store and recover a substantial amount of energy during a running gait period in their tendons and muscles. Also human running strongly relies on elastic energy storage.

Exciting periodic motions in a highly nonlinear multibody system by using nonlinear elastic actuators is, however, a nontrivial task. While modal oscillations and their control are very well understood for linear systems, the nonlinear generalization is a very active topic of research. Since we consider that the ability to perform (quasi) periodic motions with high velocity and force at low weight will be one of the major benefits of soft robots, we dedicate an extensive section to this topic.

The topic of exploiting intrinsic elasticity for periodic motions was addressed in robotics by [21.116–120] among others. Two basic approaches can be distinguished:

- Tracking an *arbitrary* desired periodic trajectory, or enforcing an arbitrary oscillatory closed-loop behavior, by control while simultaneously adjusting the compliance of the actuator such that the control effort is minimal.
- Exciting and sustaining *intrinsic* oscillatory dynamics of the compliantly actuated robotic system by control. Completion of a given task implies that this intrinsic oscillation modes correspond to the task by the initial mechanical design and by online adjustment of the robot configuration and of its intrinsic actuator stiffness.

Depending on the design of the compliantly actuated robotic system, the above control approaches have to cope with underactuation, model nonlinearities, and multiple-DOF structures.

21.7.1 Periodic Motion Tracking and Stiffness Adaptation

The idea of tracking a periodic trajectory in the link-side variables $\boldsymbol{q}$ with minimal torque is inspired by classical rigid-robot control. The approach [21.117] considers, for example, robotic systems with elastic elements *in parallel to the main actuators*

$$\mathbf{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + (\mathbf{C}(\boldsymbol{q},\dot{\boldsymbol{q}}) + \mathbf{D})\dot{\boldsymbol{q}} + \boldsymbol{g}(\boldsymbol{q}) + \mathbf{K}(\boldsymbol{q}-\boldsymbol{\theta}) = \boldsymbol{\tau}\ . \tag{21.23}$$

Since $\boldsymbol{\theta}$ is assumed to be constant here, the model reduces in this case to a fully actuated robot dynamics with parallel elasticity, being thus simpler than (21.6).

The objectives of the controller are:

1. Tracking a periodic trajectory $\boldsymbol{q}_\mathrm{d}(t+T) = \boldsymbol{q}_\mathrm{d}(t)$ with period $\boldsymbol{T}$ such that $\boldsymbol{q}(t) \to \boldsymbol{q}_\mathrm{d}(t)$
2. Minimizing the actuator torque $\boldsymbol{\tau}$ by optimizing the stiffness $\mathbf{K}$.

Considering the actuator torque required to generate the desired motion

$$\boldsymbol{\tau}_\mathrm{d} = \mathbf{M}(\boldsymbol{q}_\mathrm{d})\ddot{\boldsymbol{q}}_\mathrm{d} + [\mathbf{C}(\boldsymbol{q}_\mathrm{d},\dot{\boldsymbol{q}}_\mathrm{d}) + \mathbf{D}]\,\dot{\boldsymbol{q}}_\mathrm{d} + \boldsymbol{g}(\boldsymbol{q}_\mathrm{d}) + \mathbf{K}(\boldsymbol{q}_\mathrm{d}-\boldsymbol{\theta})\ ,$$

a stiffness matrix $\mathbf{K}_\mathrm{opt}$ is defined to be optimal if it minimizes the cost function

$$\mathbf{J}(\mathbf{K}) = \int_{i\mathrm{T}}^{i\mathrm{T}+\mathrm{T}} \boldsymbol{\tau}_\mathrm{d}(\mathbf{K},t)^\mathrm{T}\boldsymbol{\tau}_\mathrm{d}(\mathbf{K},t)\,\mathrm{d}t\ . \tag{21.24}$$

In the case of the single-DOF system

$$m\ddot{q} + d\dot{q} + kq = \tau\ , \tag{21.25}$$

with constant system parameters m, d, $k>0$, the desired, sinusoidal motion $q_\mathrm{d}(t) = a\sin\omega t$ obviously minimizes the cost function (21.24), when the stiffness is $k_\mathrm{opt} = m\omega^2$. With the optimal stiffness, the sinusoidal motion with an arbitrary amplitude $a>0$ corresponds to a resonance motion, since the angular frequency $\omega = \sqrt{k_\mathrm{opt}/m}$ of the desired trajectory corresponds to the resonance frequency of the system (21.25).

The method proposed in [21.117] extends this simple concept of resonance to the multi-DOF system (21.23). Various online or offline optimization algorithms can be defined to minimize the cost function (21.24). For example, [21.117] uses the parameter adjustment law

$$\dot{\boldsymbol{k}} = \boldsymbol{\Gamma}\,\mathrm{diag}(\boldsymbol{q}-\boldsymbol{\theta})\left[\ddot{\tilde{\boldsymbol{q}}} + \mathbf{K}_\mathrm{B}\boldsymbol{s}(\tilde{\boldsymbol{q}})\right]$$

to adapt the stiffness $\mathbf{K} = \mathrm{diag}(k)$. The controller gain matrix $\mathbf{K}_\mathrm{B}$ is positive definite and diagonal and the adaptation gain $\boldsymbol{\Gamma}$ is a positive definite matrix. The rationale behind this law is to increase the stiffness whenever the elastic deflection $(\boldsymbol{\theta}-\boldsymbol{q})$ has the opposite sign to the tracking error. In this case, a stiffness increase reduces the tracking error. The stiffness is reduced whenever the elastic torque increases the tracking

Avidbots Ex1016
Page 558 of 2154

error. In particular, the control error $\tilde{\boldsymbol{q}} = \boldsymbol{q} - \boldsymbol{q}_\mathrm{d}$ is saturated by functions $s_\mathrm{i}(\tilde{q}_\mathrm{i})$ satisfying certain boundedness constraints. These saturated functions are important for the stability analysis in [21.117].

The paper [21.121] considers dynamic systems with elastic elements either in parallel with the main actuator as in (21.23), or in series as in

$$\boldsymbol{f}(\ddot{\boldsymbol{q}}, \dot{\boldsymbol{q}}, \boldsymbol{q}, t) = -\mathbf{K}(\boldsymbol{q} - \boldsymbol{\theta}) \tag{21.26}$$

$$\mathbf{B}\ddot{\boldsymbol{\theta}} = \mathbf{K}(\boldsymbol{q} - \boldsymbol{\theta}) + \boldsymbol{\tau}\,, \tag{21.27}$$

where $\mathbf{B}$ represents the constant inertia matrix of the main motor part of the model. The objectives of [21.117] were extended to compute the optimal value of a constant stiffness for minimizing energy consumption in a periodic task for series configuration, and both the optimal stiffness and spring preload values for the parallel case.

Assuming that $\mathbf{K} = \mathrm{diag}(K_\mathrm{j})$ and $\mathbf{B} = \mathrm{diag}(B_\mathrm{j})$ (the second assumption is required in the serial case) it is shown how to derive expressions of the optimal actuation parameters for minimizing energy-related cost functions such as (21.24) and

$$\mathbf{J}(\mathbf{K}) = \int_{\mathrm{iT}}^{\mathrm{iT}+\mathrm{T}} \boldsymbol{w}_\mathrm{d}(\mathbf{K}, t)^\mathrm{T} \boldsymbol{w}_\mathrm{d}(\mathbf{K}, t)\,\mathrm{d}t\,. \tag{21.28}$$

where $\boldsymbol{w}_\mathrm{d}(\mathbf{K}, t) = \boldsymbol{\tau}_\mathrm{d}(\mathbf{K}, t) \cdot^* \dot{\boldsymbol{\theta}}_\mathrm{d}(\mathbf{K}, t)$, with $\cdot^*$ denoting element-wise multiplication. Rather surprisingly, it is shown in [21.121] that it is possible to save lengthy iterations and simulations by computing analytical expressions for the optimal stiffness and preload as integral functions of any given desired joint trajectories: for example, the series stiffness that minimizes the cost (21.24) is given by the formula

$$\mathbf{K}_\mathrm{j}^* = -\frac{\int_{i\mathrm{T}}^{i\mathrm{T}+\mathrm{T}} \mathbf{B}_\mathrm{j}^2 \ddot{f}_\mathrm{j}(\ddot{\boldsymbol{q}}_\mathrm{d}, \dot{\boldsymbol{q}}_\mathrm{d}, \boldsymbol{q}_\mathrm{d}, t)\,\mathrm{d}t}{\int_{i\mathrm{T}}^{i\mathrm{T}+\mathrm{T}} \mathbf{B}_j \ddot{f}_j(\ddot{\boldsymbol{q}}_\mathrm{d}, \dot{\boldsymbol{q}}_\mathrm{d}, \boldsymbol{q}_\mathrm{d}, t)(\mathbf{B}_j \ddot{q}_{\mathrm{d},j} + f_j(\ddot{\boldsymbol{q}}_\mathrm{d}, \dot{\boldsymbol{q}}_\mathrm{d}, \boldsymbol{q}_\mathrm{d}, t))\,\mathrm{d}t}\,. \tag{21.29}$$

21.7.2 Controlled Oscillatory Dynamics

The basic idea to generate periodic motions is here to control the output of the joint such that it behaves like a desired, oscillatory dynamic system rather than imposing a predefined, time-dependent trajectory [21.118]. The resulting motion will be therefore depending on initial conditions and possible perturbations, but will eventually converge to the periodic orbit. For simplicity, here we describe the concept for the model of a single variable stiffness actuated joint, satisfying

$$m\ddot{q} + \frac{\partial V_\tau(\theta, q)}{\partial q} = 0\,. \tag{21.30}$$

The output position of the joint is denoted $q \in \mathbb{R}$ and the velocities $\dot{\theta} = u \in \mathbb{R}^2$ of the two actuators of the VIA joint are assumed as control inputs. Note that this model corresponds to the first row in (21.9) with $m = 2$ and $n = 1$. Let us assume a generic family of desired oscillatory dynamics

$$\ddot{q} + a(q) = 0\,, \tag{21.31}$$

with $a(q)$ chosen such that a periodic orbit results. In particular, $a(q)$ can be the derivative of a desired potential function $V_\mathrm{d}(q)$. The goal is to control the system (21.30) such that

$$h(\theta, q) = a(q) - \frac{1}{m}\frac{\partial V_\tau(\theta, q)}{\partial q} = 0 \tag{21.32}$$

holds. With initial conditions $(q(0), \dot{q}(0), \theta_1(0), \theta_2(0)) \in h(0)^{-1}$ the constraint (21.32) can be fulfilled, if the control u satisfies the constraint on velocity level, that is, $\dot{h}(\theta, q) = 0$. Since $\dot{\theta} = u$ is the control input, this implies

$$\dot{h}(\theta, q) = -\mathbf{A}(\theta, q)u + b(\theta, q, \dot{q}) = 0\,, \tag{21.33}$$

with

$$\mathbf{A}(\theta, q) := \frac{1}{m}\frac{\partial^2 V_\tau(\theta, q)}{\partial q \partial \theta}\,, \tag{21.34}$$

$$b(\theta, q, \dot{q}) := \left(\frac{\partial a(q)}{\partial q} - \frac{1}{m}\frac{\partial^2 V_\tau(\theta, q)}{\partial q^2}\right)\dot{q}\,. \tag{21.35}$$

Using a generalized right inverse $\mathbf{A}^\dagger$ of the matrix $\mathbf{A}$, the control

$$u = \mathbf{A}(\theta, q)^\dagger b(\theta, q, \dot{q})\,, \tag{21.36}$$

fulfills the velocity constraint (21.33).

The solution of (21.33) with regard to u is not unique, as the problem is under-constrained. By choosing a particular generalized pseudoinverse of $\mathbf{A}$ of the form $\mathbf{A}^\dagger = \mathbf{B}^{-1}\mathbf{A}^\mathrm{T}(\mathbf{A}\mathbf{B}^{-1}\mathbf{A}^\mathrm{T})^{-1}$, the solution (21.36) minimizes the norm $\|u\|_\mathrm{B}^2$ with respect to a certain metric tensor $\mathbf{B}$. If, for example, $\mathbf{B}$ is the matrix containing the motor inertias then the kinetic energy of the actuators is minimized. The under-constrained situation can be interpreted such that the desired motion might be achieved by both adjusting the stiffness and the motor position of the actuators. Choosing a certain metric $\mathbf{B}$ implicitly defines a weighting of the two components.

Avidbots Ex1016
Page 559 of 2154

21.7.3 Exciting Intrinsic Oscillatory Dynamics

Exciting the intrinsic oscillatory behavior of mechanical systems, that consist of kinetic and potential energy storages, is inspired by the concept of resonance from linear oscillation theory. The idea is to move the actuators of the compliant system such that the trajectory of link variables is periodic and thereby the energy input due to the motor motion ideally counteracts only the loss of dissipated energy. Instead of tracking an arbitrary trajectory, here we are interested in exciting intrinsic resonant motions of the system.

Consider the dynamical system of the form (21.10). The objectives in this section are finding a control $\boldsymbol{\tau}_m$ such that for a certain period T, the trajectories of link variables fulfill $\boldsymbol{q}(t+T) = \boldsymbol{q}(t)$, and the intrinsic oscillatory dynamics are exploited to the maximally possible extent. Assuming that the motion $\boldsymbol{q}(t)$ should take place on one of several one-dimensional manifolds $\mathcal{Z}_j \subset \mathbb{R}^n$ (referred to as oscillation modes), the problem can be further refined:

- *C1*: Given the mapping $z = z(\boldsymbol{q})$. If z_j represents the coordinate of the desired oscillation mode, then the control must ensure that oscillations excited in all other modes decay, that is, for $i \neq j$, $\dot{z}_i \to 0$.
- *C2*: The motion of z_j has to be controlled to sustain a periodic motion with a desired oscillation amplitude.

21.7.4 Model-Based Approach

To derive a coordinate transformation $z = z(\boldsymbol{q})$ and to fulfill the above conditions, the free motion ($\boldsymbol{\tau}_{\text{ext}} = \boldsymbol{0}$) link-side dynamics of (21.10) can be modified by first considering $\boldsymbol{\theta}$ as control input, that is,

$$\boldsymbol{\theta}_{\text{d}} = \boldsymbol{q}^{\star} + \mathbf{K}_{\text{P}}^{-1}(\mathbf{C}(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{g}(\boldsymbol{q}) - \mathbf{K}_{\text{D}}\dot{\boldsymbol{q}} + \boldsymbol{\gamma}) \,, \tag{21.37}$$

with $\boldsymbol{\gamma}$ being a decoupling term to be specified later. This leads to the closed-loop dynamics

$$\mathbf{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \mathbf{K}_{\text{P}}(\boldsymbol{q} - \boldsymbol{q}^{\star}) + \mathbf{K}_{\text{D}}\dot{\boldsymbol{q}} = 0 \,. \tag{21.38}$$

In order to excite oscillations which are quite close to the initial oscillation modes, consider coordinates z_{i} which correspond to the instantaneous modal coordinates $z = \mathbf{Q}(\boldsymbol{q})^{-1}(\boldsymbol{q} - \boldsymbol{q}^{\star})$. $\mathbf{Q}(\boldsymbol{q})$ is defined on the basis of double diagonalization of $\mathbf{M}$ and $\mathbf{K}$ such that $\mathbf{Q}(\boldsymbol{q})^{-\text{T}}\mathbf{Q}(\boldsymbol{q})^{-1} = \mathbf{M}(\boldsymbol{q})$ and $\mathbf{Q}(\boldsymbol{q})^{-\text{T}}\text{diag}(\lambda(\boldsymbol{q}))\mathbf{Q}(\boldsymbol{q})^{-1} = \mathbf{K}_{\text{P}}$, where $\boldsymbol{\lambda}(\boldsymbol{q}) \in \mathbb{R}^n_{>0}$ are generalized eigenvalues, as proposed in [21.120]. By considering a damping design

$$\mathbf{K}_{\text{D}} = 2\mathbf{Q}(\boldsymbol{q})^{-\text{T}}\text{diag}\left[\xi_{\text{i}}\sqrt{\lambda_{\text{i}}(\boldsymbol{q})}\right]\mathbf{Q}(\boldsymbol{q})^{-1} \,,$$

and by choosing γ as

$$\boldsymbol{\gamma} = [\mathbf{M}(\boldsymbol{q})\ddot{\mathbf{Q}}(\boldsymbol{q}) + K_{\text{D}}\dot{\mathbf{Q}}(\boldsymbol{q})]z + 2\mathbf{M}(\boldsymbol{q})\dot{\mathbf{Q}}(\boldsymbol{q})\dot{z} \,, \tag{21.39}$$

the resulting closed-loop dynamics

$$\ddot{z}_i + 2\xi_i\sqrt{\lambda_i(\boldsymbol{q})}\dot{z}_i + \lambda_i(\boldsymbol{q})z_i = 0$$

fulfills the condition *C1*, if for $i \neq j$, $\xi_i > 0$. Then, the nonlinear damping term

$$\xi_j = -\frac{\dot{\lambda}_j(\boldsymbol{q})}{4\lambda_j(\boldsymbol{q})^{\frac{3}{2}}} + \frac{1}{2}\sqrt{\lambda_j(\boldsymbol{q})}k_{\text{H}}[H(\boldsymbol{q}, z_j, \dot{z}_j) - H_{\text{d}}]$$

produces a periodic motion $z_{\text{j}}(t+T) = z_{\text{j}}(t)$. In particular, the resulting limit cycle corresponds to a level H_{d} of the energy function $H(\boldsymbol{q}, z_j, \dot{z}_j) = \frac{1}{2}(\frac{1}{\lambda_j(\boldsymbol{q})}\dot{z}_j^2 + z_j^2)$ as proposed in [21.122].

Note that applying the control (21.37) to the system (21.10) requires a controller that ensures a tracking $\boldsymbol{\theta}(t) \to \boldsymbol{\theta}_{\text{d}}(t)$ of the desired motor position $\boldsymbol{\theta}_{\text{d}}(t)$.

21.7.5 Model-Free Approach

Exciting oscillations in multi-DOF mechanical systems without changing at all the original inertia and stiffness properties of the plant requires either an ideally separated excitation on a certain oscillation mode or intrinsic physical damping properties that ensure the decay of oscillations excited in the other modes cf. condition *C1*. Note that in most cases fully decoupled modes do not even exist. On the basis of the latter assumption, which is valid in biological systems and some robotic examples [21.123], the control law

$$\Delta\boldsymbol{\theta}_z = \begin{cases} \text{sign}(\tau_z)\hat{\theta}_z & \text{if } |\tau_z| > \epsilon_{\tau_z} \\ 0 & \text{otherwise} \end{cases} \tag{21.40}$$

excites a modal oscillation in a simple way. Thereby, the linear transformation $z_{\text{j}} = \boldsymbol{w}^{\text{T}}\boldsymbol{q}$ approximates the nonlinear mapping $\varphi\colon \mathbb{R}^n \to \mathcal{Z}_j$, such that the generalized modal force can be transformed by the vector of weights $\boldsymbol{w} \in \mathbb{R}^{\text{n}}$, that is,

$$\tau_z = \boldsymbol{w}^{\text{T}}(\boldsymbol{\psi}(\boldsymbol{\theta} - \boldsymbol{q}) - \boldsymbol{\psi}(\boldsymbol{\theta}^{\star} - \boldsymbol{q}^{\star})) \,, \tag{21.41}$$

where $\boldsymbol{\theta}^{\star}$ and $\boldsymbol{q}^{\star}$ satisfy the equilibrium conditions $\boldsymbol{g}(\boldsymbol{q}) + \boldsymbol{\psi}(\boldsymbol{q} - \boldsymbol{\theta}) = 0$. When the modal force τ_{z} exceeds a certain threshold ϵ_{τ_z}, the controller switches by an

amplitude $\hat{\boldsymbol{\theta}}_z$. This results in steps of desired motor positions

$$\boldsymbol{\theta}_d = \boldsymbol{\theta}^\star + \boldsymbol{w}\Delta\boldsymbol{\theta}_z \,.$$

Since the motor positions θ are not the control input of the system (21.10), a motor PD controller should be considered to ensure that $\boldsymbol{\theta}(t) \to \boldsymbol{\theta}_d(t)$.

Note that the oscillation controller requires neither derivatives of measured signals nor model knowledge of the plant. Therefore it is simple and robust, and can be easily applied to different types of compliantly actuated systems. The control law is, at least for the one-DOF case, the optimal solution for maximizing the oscillation amplitude in minimum time. Furthermore, the control law defined by (21.40) and (21.41) is plausible to have a biological equivalent, since it strongly resembles the computation function of neurons.

21.8 Optimal Control of Soft Robots

Robot joints with intrinsic elasticity have some major advantages over their rigid counterparts: they are more robust, versatile, and can store energy. These properties enable robots to realize motions which would not be possible using stiff actuators of the same size. However, the question remains open as to how elastic elements in soft robots can be best controlled to exploit their potential. Optimal control (OC) theory is being regarded as one of the most promising tools to clarify such questions: as far as an appropriate choice of the cost functional is available, optimal control techniques can provide design guidelines for both the physical design and the control of soft robots.

There are at least two main reasons why OC is fundamental to study soft actuators. On one side OC provides an absolute performance reference that factorizes the control design out of the equation, hence it furnishes a principled basis to compare the performance of different system designs. On the other side, OC is a key element in understanding planning and control methodologies for soft actuators. A careful analysis of OC results obtained through either analytic or numerical techniques, allows to distillate laws summarizing control policies that can be applied to classes of tasks.

General analytic solutions could be developed so far only for one-DOF joints. Physical bounds on states and inputs, the strong nonlinear structure of the robots and the high number of DOF allow only numerical solutions for general robotic systems. Nevertheless, choosing an advantageous parametrization for numerical solutions of the OC problem strongly benefits from insights given by the low-dimensional analytical solutions. Therefore, the analytical concepts are explained for some basic examples in this section.

21.8.1 OC Theory

We consider the dynamics of a system described in standard form ($\dot{x} = f(u, x, t)$), with the cost functional J consisting of a terminal cost ϑ and an integral cost L

$$J(u) = \vartheta(x(t_\mathrm{f}), t_\mathrm{f}) + \int_{t_0}^{t_\mathrm{f}} L(x, u, t)\,\mathrm{d}t \,. \tag{21.42}$$

According to Pontryagin's minimum principle [21.124], the optimal control u^* is known to minimize the Hamiltonian function $\mathcal{H} = \lambda^\mathrm{T} f(u, x, t) + L$, along the optimal trajectory, where λ are the costates. For an OC problem with constraints on only the control $u \in \mathbb{U}$, differential equations of these costates are given by $\dot{\lambda} = -\partial\mathbb{H}/\partial x$. Furthermore, if the final state is not explicitly stated, Minimum Principle also provides boundary conditions for $\boldsymbol{\lambda}$ at the final time

$$\lambda(t_\mathrm{f}) = \left.\frac{\partial\vartheta}{\partial x}\right|_{t_\mathrm{f}} \,.$$

In order to make use of the Minimum Principle, one has to first solve these differential equations for x and λ and then find the control u^*, which minimizes the Hamiltonian: $\mathbb{H}(x^*, \lambda^*, u^*) \leq \mathbb{H}(x^*, \lambda^*, u)$ for all $u \in \mathbb{U}$, where x^* and λ^* denote the optimal states and costates.

21.8.2 SEA Optimization

Naturally, the performance achievable with a given soft actuator under optimal control depends on the physics of the actuator itself. It is therefore very interesting to study the nested optimization problem of how to design soft robotics actuators which can produce the best performance under optimal control.

From the analysis of a one-DOF SEA joint, it can be observed [21.17] that there exists an optimal value of the linear stiffness that maximizes the (optimally controllable) peak speed. The optimal stiffness value depends on the motor (e.g., maximum acceleration and speed) and on the task (e.g., terminal time). This result is illustrated in Fig. 21.12.

Avidbots Ex1016
Page 561 of 2154

21.8.3 Safe Brachistochrone OC Problem

One of the first instances of use of OC for soft (variable stiffness) robots is the study of the so-called *safe brachistochrone* problem [21.125]. The goal was to find control laws for the equilibrium position and variables transmission stiffness such that a point-to-point link motion is achieved in minimum time subject to a limit on the worst-case impact force that the link could exchange with the environment during the trajectory. Limiting the impact force during motion can be useful both for increasing the safety of a robot operating in proximity of humans, and for reducing the risk of damaging the robot itself in accidental collisions with a stiff environment. The main results were to show that VSAs can perform better (that is, faster within bounds on a measure of impact risk) than both conventional (rigid) actuators and SEAs, and that the optimal stiffness law is inversely related to the link velocity, in a fashion roughly described by the easy-to-remember rule *fast and soft, stiff and slow*.

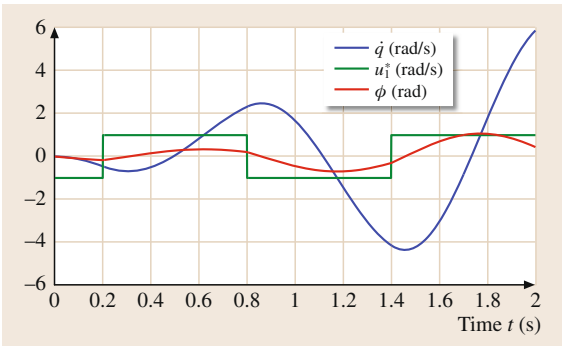


Fig. 21.12 Optimal motor velocity ($\omega = 5\,\mathrm{rad/s}$, $|u_{1\,\min}| = u_{1\,\max} = 1\,\mathrm{rad/s}$)

21.8.4 Max Speed OC Problem

A complementary example of application of OC theory concerns the problem of maximizing the velocity of an elastic robot joint at some desired final time or position. This goal is relevant to impulsive tasks, such as jumping on feet, hitting a ball or hammering a nail. Conversely to the previous example, in such tasks it might be desirable that impact is maximized: indeed, it turns out that elastic actuators are neither intrinsically more or less safe than rigid joints, but can be both, under suitable control laws.

The differential equations for the states and costates of this OC problem are summarized in Table 21.1. Note that the functional J to be minimized consists merely of a terminal cost $J(u_1) = -\dot{q}(t_\mathrm{f})$. For this particular problem, studied in [21.126, 127], the condition of minimizing the Hamiltonian $\mathbb{H}$ on the optimal trajectory becomes $u_1^*\lambda_1^* \le u_1\lambda_1^*$, for all $u_1 \in [u_{1\,\min}, u_{1\,\max}]$. Consequently, u_1^* will have its minimum or maximum value depending on the sign of the first costate λ_1^*. Using Table 21.1, this first costate can be obtained as $\lambda_1^* = -\omega \sin[\omega(t_\mathrm{f} - t)]$, where ω denotes the natural frequency of the system. The optimal control u_1^* will thus be periodic with the eigenfrequency of the system ω and take its maximum (minimum) value, whenever $\sin[\omega(t_\mathrm{f} - t)]$ is positive (negative). Note that the switching of the optimal control occurs with the eigenfrequency starting from the final time backward. For the particular case

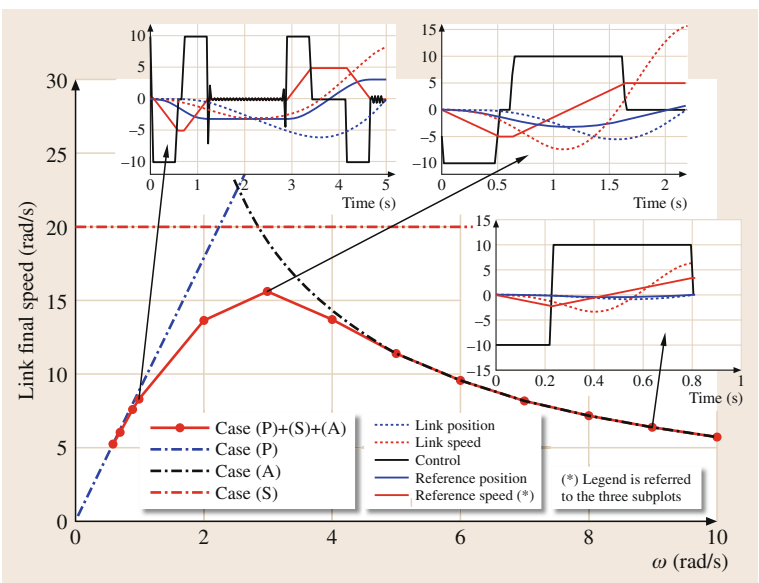


Fig. 21.13 Diagram (after [21.17]) representing the maximum speed reached by a one-DOF SEA after a single swing at varying the spring stiffness ($\omega = \sqrt{k/m}$). *Dashed curves* show terminal speed $\dot{q}(T)$ under ideal motor position control (case P), speed control (case S) and acceleration control (case A). *The continuous red curve* shows terminal speed in the realistic condition of motor acceleration control ($|\ddot{\theta}(t)| \le 10\,\mathrm{rad\,s^{-2}}$), with bounds on position ($|\theta(t)| \le \pi$ rad) and velocity ($|\dot{\theta}(t)| \le 5\,\mathrm{rad\,s^{-1}}$). The *small charts* show optimal control and state patterns in time

Avidbots Ex1016
Page 562 of 2154

Table 21.1 OC problem for the elastic joint ($\omega = \sqrt{\frac{k}{m}}$)

Excitation	States x	Cost J	$\dot{x} = f(x, u_1)$	$\dot{\lambda} = -\frac{\partial \mathbb{H}}{\partial x}$	$\lambda(t_f)$
$u_1 = \dot{\theta}$	$\begin{pmatrix}\phi\\ \dot{q}\end{pmatrix}$	$-x_2(t_f)$	$\begin{pmatrix}u_1 - x_2\\ \omega^2 x_1\end{pmatrix}$	$\begin{pmatrix}-\omega^2\lambda_2\\ \lambda_1\end{pmatrix}$	$\frac{\partial \vartheta}{\partial x}\|_{t_f} = \begin{pmatrix}0\\ -1\end{pmatrix}$

when the final time is a multiple of the oscillation period $t_f = nT = 2\pi n/\omega$, switching can be equivalently done in feed-forward after every half period. For an elastic joint starting from the rest position, this particular control law states then to switch the motor velocity when the angular deflection ϕ or similarly the acceleration of the system changes its sign

$$\dot{\theta}^* = \text{sign}(\ddot{q}) u_{1\,\max} \,. \tag{21.43}$$

If an additional quadratic integral cost in the control, $L = cu_1^2$ with a positive scalar c, is considered in (21.42), a continuous control law will follow. For $t_f = nT$ a classical resonance excitation with a sinusoidal input then follows, corresponding to intuition and knowledge of linear systems. It is possible to extend these analytical solutions for nonlinear stiffness characteristics and damped systems [21.127]. For example, for an underdamped robotic link a similar optimal control is found, which is periodic with the damped eigenfrequency. Different motor models can also be investigated analytically.

21.8.5 VSA Optimization

The very fact that for different tasks the optimal value of stiffness is different implies that the adaptability of the physical stiffness of actuators can be very useful in improving performance over a set of tasks.

One further interesting question is raised by the possibility offered by VSA technologies to vary the actuator stiffness in time. The problem becomes that of finding the optimal time course of both the actuator position $\theta(t)$ and stiffness $k(t)$ so as to maximize performance. This problem can be again investigated using OC theory [21.17, 128]. Without going in the details of OC computations, results show that under the simplifying assumption that the actuator position and stiffness are bounded but can be changed instantaneously, the optimal control for a single swing and free terminal time (within this swing) is bang–bang: the optimal reference position is

$$\theta^* = \text{sign}(\dot{q})\theta_{\max} \,,$$

while the optimal stiffness $k^*(t)$ takes its minimum value $k_{\min}$ or maximum value $k_{\max}$ depending on the switching function

$$k^* = \begin{cases} k_{\min} & \text{if } \dot{q}\ddot{q} < 0 \\ k_{\max} & \text{if } \dot{q}\ddot{q} > 0 \end{cases} , \tag{21.44}$$

while for $\dot{q}\ddot{q} = 0$ the stiffness value can be set arbitrarily. This indicates the rather intuitive notion between link velocity and accelerations summarized by

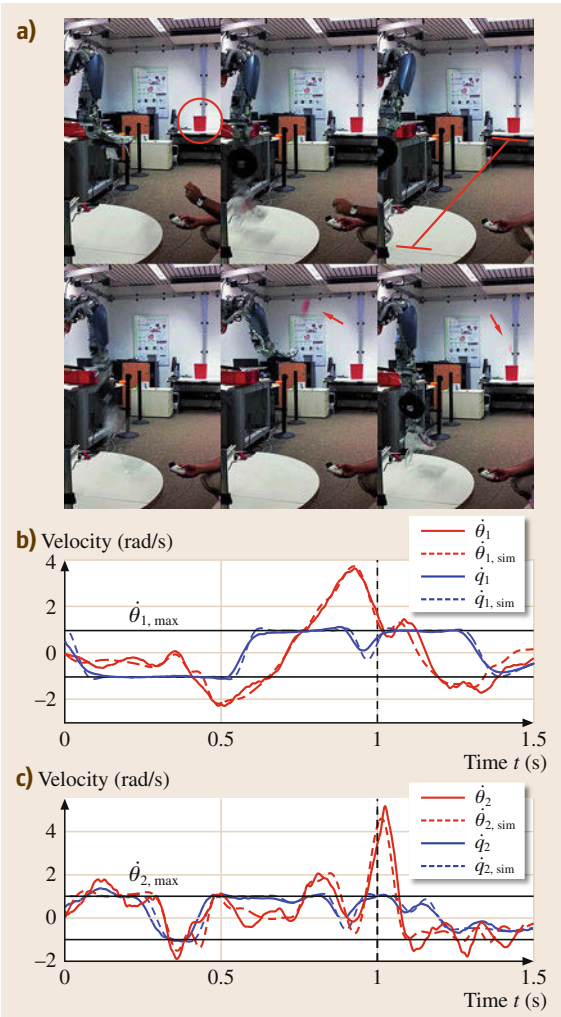


Fig.21.14a–c Throwing experiment with the DLR hand-arm system (after [21.87]). **(a)** Experimental setup, **(b)** velocity of joint 1, **(c)** velocity of joint 2

Avidbots Ex1016
Page 563 of 2154

the rule *stiff speed-up, soft slow-down*. More general cases of VSA optimization are discussed in [21.17, 128] and [21.126]. It is important to notice that even for one-link robots analytical solutions are not always straightforward to find, especially if state constraints exist or system properties have nonlinearities.

Under some conditions, it may be possible to apply efficient numerical solutions to such problems. For instance, if the soft actuator dynamics system are linear (as e.g., in the SEA case) but there exist state and control constraints (e.g., the torque-speed characteristic of a dc motor), it is possible to translate the optimal control problem into a convex optimization problem [21.129], for which there exist fast and reliable methods to obtain the global solution. For robots with multiple-links, differential equations resulting from OC theory are nonlinear and cannot in general be solved analytically. Consequently, numerical methods such as pseudospectral methods [21.130] or iterative linear quadratic regulator (ILQR) methods [21.131, 132] are employed for these systems. One example of how to use these methods to outperform rigid robots is described in [21.133], where a throwing motion is optimized using the gauss pseudospectral method [21.134]. The reported experiments verify the system's ability to use the elastic energy for explosive motions. Figure 21.14a illustrates the experimental setup, while the motor and link velocity are depicted on Fig. 21.14b,c. Note that the link velocities $\dot{q}$ reach more than double the maximum motor velocities $\dot{\theta}_{\max}$ during the motion, which would be clearly not possible with stiff actuators.

21.9 Conclusions and Open Problems

In this chapter, we have reviewed some of the technologies and methods for analysis and control underpinning the use of novel actuators for soft robotics. The field has rapidly expanded in recent years, and it is very difficult to provide an exhaustive up-to-date account of all innovations being continuously brought forward: hopefully, this chapter provides at least a faithful picture of some of the most important current research trends in soft actuation.

When comparing current robots with biological systems in terms of actuator energy density, energy efficiency of motion, resilience, and sensitivity during interaction, the importance of intrinsic compliance becomes striking. We are still far from having found so far the optimal technical realization of compliant, muscle-like actuators regarding design simplicity, integration, and energy density. Nevertheless, in this chapter foundations have been described both by properly defining basic notions such as nonlinear impedance and addressing parametric identification, and in analyzing and standardizing the main physical properties to be used for the description of elastic actuators. A widely agreed set of characteristic parameters summarized in a standardized data sheet format [21.75–77] help comparing the very heterogeneous design approaches and choosing the best concept for a given application. We proposed a quite general dynamic model for flexible robots with lumped elasticities, covering all designs proposed to our knowledge so far. The model is obviously related to the models of flexible robots from Chap. 11 and can be regarded as a generalization thereof. The control of flexible robots was focused on two main challenges. First, the simultaneous control of the position and intrinsic impedance of the devices has to be solved. This is a challenge for systems with highly nonlinear and very compliant behavior. This aspect was addressed on joint and the Cartesian level.

Second, control to push flexible robots to their limit and maximally exploit their potential benefits. If robustness comes mainly by the design process, that is not true for energy efficiency and performances maximization (even under safety constraints). The achievement of these features requires optimization and optimal control algorithms, such as those exposed in the present chapter. Moreover, OC algorithms can be used as a tool for comparison among different actuator designs extracting their inherent (i. e., control-independent) performance boundaries, and to derive easy-to-understand control rules for peculiar DOFs (e.g., stiffness) of soft robots.

Important new directions are open in front of the research community. On one side, new active materials could provide a boost to the development in soft robotics actuators, especially in the small scale; on the other hand, it can be expected that some of the promising advanced material research in the past will eventually make their way in practically viable technologies. New functional architectures for actuators that can adapt their physical behavior to the task are also potentially very fruitful. An important future trend is the extension from *variable stiffness* to VIAs, capable of more closely mimicking the functionalities of muscular structures where not just the elasticity, but also the damping (and perhaps the effective inertia) can be physically tuned. Much work remains also to be done in the understanding of the theoretical underpinnings and in practical methods for the identification and

Avidbots Ex1016
Page 564 of 2154

control of nonlinear, time varying, multidimensional impedance.

The very fact that research in novel actuators for soft robotics has produced a wealth of ideas and solutions poses a challenge of different nature. Indeed, a huge potential for innovation is already available to researchers and practitioners, which has been only superficially tapped so far in applications. Soft actuators could change the landscape not only in robust and safe interaction or in locomotion: there are probably many, more specific application niches where the new technologies could have an impact – possibly also outside the scope of *robotics* narrowly defined, in the larger domain if *automation*. To foster technological awareness of soft robotics potential in the community of those interested in machines that can move and interact more naturally, several initiatives are being undertaken, including an IEEE technical committee on soft robotics [21.135] and several workshops, special issues and even specialized journals on the topic. Recently, a natural machine motion initiative (NMMI) [21.136] has been launched with the aim of diffusing the ideas of soft actuation to achieve *naturally moving* robots with human-like grace, power, and dexterity. One of the objectives of the NMMI is to make the technology of soft actuators widely available to the general public, by publishing as open-source the design and all instructions needed to build and use low-cost actuators, thus building a community of developers interested in sharing advances toward simple, affordable, yet smooth, strong, and accurate muscles for robots.

Video-References

VIDEO 456 Variable impedance actuators: Moving the robots of tomorrow
available from http://handbookofrobotics.org/view-chapter/21/videodetails/456

VIDEO 457 Petman tests camo
available from http://handbookofrobotics.org/view-chapter/21/videodetails/457

VIDEO 458 Introducing WildCat
available from http://handbookofrobotics.org/view-chapter/21/videodetails/458

VIDEO 460 VSA CubeBot – peg in hole
available from http://handbookofrobotics.org/view-chapter/21/videodetails/460

VIDEO 461 DLR Hand Arm System smashed with baseball bat
available from http://handbookofrobotics.org/view-chapter/21/videodetails/461

VIDEO 463 Safety evaluation of lightweight robots
available from http://handbookofrobotics.org/view-chapter/21/videodetails/463

VIDEO 464 Hammering task with the DLR Hand Arm System
available from http://handbookofrobotics.org/view-chapter/21/videodetails/464

VIDEO 465 Dynamic walking of whole-body compliant humanoid COMAN
available from http://handbookofrobotics.org/view-chapter/21/videodetails/465

VIDEO 466 Dynamic walking of whole-body compliant humanoid COMAN
available from http://handbookofrobotics.org/view-chapter/21/videodetails/466

VIDEO 467 Maccepa System
available from http://handbookofrobotics.org/view-chapter/21/videodetails/467

VIDEO 468 AMASC – changing stiffness
available from http://handbookofrobotics.org/view-chapter/21/videodetails/468

VIDEO 470 VSA-Cube: Arm with high and low stiffness preset
available from http://handbookofrobotics.org/view-chapter/21/videodetails/470

VIDEO 471 CompAct™ robotics technology
available from http://handbookofrobotics.org/view-chapter/21/videodetails/471

VIDEO 472 VSA-Cube arm: Drawing on a wavy surface (high stiffness)
available from http://handbookofrobotics.org/view-chapter/21/videodetails/472

VIDEO 473 Arm drawing on a wavy surface (low stiffness)
available from http://handbookofrobotics.org/view-chapter/21/videodetails/473

VIDEO 474 Arm drawing on a wavy surface (selective stiffness)
available from http://handbookofrobotics.org/view-chapter/21/videodetails/474

VIDEO 475 Intrinsically elastic robots: The key to human like performance (Best Video Award)
available from http://handbookofrobotics.org/view-chapter/21/videodetails/475

VIDEO 546 DLR Hand Arm System: Punching holes
available from http://handbookofrobotics.org/view-chapter/21/videodetails/546

VIDEO 547 DLR Hand Arm System throwing a ball and Justin catching it
available from http://handbookofrobotics.org/view-chapter/21/videodetails/547

Avidbots Ex1016
Page 565 of 2154

VIDEO 548 Active damping control on the DLR Hand Arm System
available from http://handbookofrobotics.org/view-chapter/21/videodetails/548
VIDEO 549 Throwing a ball with the DLR VS-Joint
available from http://handbookofrobotics.org/view-chapter/21/videodetails/549
VIDEO 550 DLR Hand Arm System: Two arm manipulation
available from http://handbookofrobotics.org/view-chapter/21/videodetails/550
VIDEO 698 Full body compliant humanoid COMAN
available from http://handbookofrobotics.org/view-chapter/21/videodetails/698
VIDEO 699 AWAS-II
available from http://handbookofrobotics.org/view-chapter/21/videodetails/699

References

21.1 K.T. Ulrich, T.T. Tuttle, J.P. Donoghue, W.T. Townsend: *Intrinsically Safer Robots*, *Tech. Rep.* (Barrett Technology Inc., Cambridge 1995), Final Report for NASA Contract NAS10-12178

21.2 B. Lacevic, P. Rocco: Safety-oriented control of robotic manipulators – A kinematic approach, Proc. 18th World Congr. Int. Fed. Autom. Control (IFAC) (2011)

21.3 S. Haddadin, S. Haddadin, A. Khoury, T. Rokahr, S. Parusel, R. Burgkart, A. Bicchi, A. Albu-Schaeffer: On making robots understand safety: Embedding injury knowledge into control, Int. J. Robotics Res. **31**, 1578–1602 (2012)

21.4 K. Salisbury, W. Townsend, B. Eberman, D. DiPietro: Preliminary design of a whole-arm manipulation system (WAMS), Proc. IEEE Int. Conf. Robotics Autom. (ICRA), Philadelphia (1988) pp. 254–260

21.5 G. Hirzinger, A. Albu-Schäffer, M. Hähnle, I. Schaefer, N. Sporer: On a new generation of torque controlled light-weight robots, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (2001) pp. 3356–3363

21.6 K. Yokoi, F. Kanehiro, K. Kaneko, S. Kajita, K. Fujiwara, H. Hirukawa: Experimental study of humanoid robot HRP-1S, Int. J. Robotics Res. **23**(4/5), 351–362 (2004)

21.7 K. Kaneko, F. Kanehiro, M. Morisawa, K. Miura, S. Nakaoka, S. Kajita: Cybernetic human HRP-4C, Proc. IEEE/RAS Int. Conf. Humanoid Robots (2009) pp. 7–14

21.8 Y. Sakagami, R. Watanabe, C. Aoyama, S. Matsunaga, N. Higaki, K. Fujimura: The intelligent asimo: System overview and integration, Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS), Vol. 3 (2002) pp. 2478–2483

21.9 M. Hirose, K. Ogawa: Honda humanoid robots development, Phil. Trans. Roy. Soc. A **365**(1850), 11–19 (2007)

21.10 B. Vanderborght, B. Verrelst, R. Van Ham, M. Van Damme, D. Lefeber: A pneumatic biped: Experimental walking results and compliance adaptation experiments, Proc. Int. Conf. Humanoid Robots, Tsukuba (2006)

21.11 T. Takuma, K. Hosoda, M. Ogino, M. Asada: Stabilization of quasi-passive pneumatic muscle walker, Proc. IEEE/RAS Int. Conf. Humanoid Robots, Vol. 2 (2004) pp. 627–639

21.12 S.H. Collins, A. Ruina: A bipedal walking robot with efficient and human-like gait, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (2005) pp. 1983–1988

21.13 N.G. Tsagarakis, S. Morfey, G. Medrano-Cerda, Z. Li, D.G. Caldwell: Compliant humanoid coman: Optimal joint stiffness tuning for modal frequency control, Proc. IEEE Int. Conf. Robotics Autom. (2013) pp. 665–670

21.14 J.W. Hurst, J.E. Chestnutt, A.A. Rizzi: *An Actuator with Mechanically Adjustable Series Compliance*, CMU-RI-TR-04-24 (Robotics Inst./Carnegie Mellon University, Pittsburgh 2004)

21.15 L.C. Visser, S. Stramigioli, R. Carloni: Robust bipedal walking with variable leg stiffness, Proc. 4th IEEE/RAS/EMBS Int. Conf. Biomed. Robotics Biomechatron. (BioRob) (2012) pp. 1626–1631

21.16 S. Haddadin, M.C. Ozparpucu, A. Albu-Schäffer: Optimal control for maximizing potential energy in a variable stiffness joint, Proc. 51st IEEE Annu. Conf. Decis. Control (CDC) (2012) pp. 1199–1206

21.17 M. Garabini, A. Passaglia, F.A.W. Belo, P. Salaris, A. Bicchi: Optimality principles in variable stiffness control: The VSA hammer, Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS), San Francisco (2011) pp. 1843–1851

21.18 A.G. Feldman: Once more on the equilibrium-point hypothesis (λ model) for motor control, J. Mot. Behav. **18**, 17–54 (1986)

21.19 J.R. Flanagan, A.M. Wing: Modulation of grip force with load force during point-to-point arm movements, Exp. Brain Res. **95**, 131–143 (1993)

21.20 J. Won, N. Hogan: Stability properties of human reaching movements, Exp. Brain Res. **107**(1), 125–136 (1995)

21.21 P.L. Gribble, D.J. Ostry, V. Sanguineti, R. Laboissière: Are complex control signals required for human arm movement?, J. Neurophysiol. **79**, 1409–1424 (1998)

21.22 J.D.W. Madden, N.A. Vandesteeg, P.A. Anquetil, P.G.A. Madden, A. Takshi, R.Z. Pytel, S.R. Lafontaine, P.A. Wieringa, I.W. Hunter: Artificial muscle technology: Physical principles and naval prospects, IEEE J. Ocean. Eng. **29**(3), 706–728 (2004)

21.23 A. Parr: *Hydraulics and Pneumatics: A Technician's and Engineer's Guide* (Butterworth-Heinemann, Oxford 1998)

Avidbots Ex1016
Page 566 of 2154

21.24 I.L. Krivts, G.V. Krejnin: *Pneumatic Actuating Systems for Automatic Equipment: Structure and Design* (CRC, Boca Raton 2006)

21.25 H.A. Baldwin: Realizable models of muscle function, Proc. 1st Rock Biomech. Symp. (1969) pp. 139–148

21.26 K. Inoue: Rubbertuators and applications for robots, Proc. 4th Int. Symp. Robotics Res. (1987) pp. 57–63

21.27 B. Hannaford, J.M. Winters, C.-P. Chou, P.H. Marbot: The anthroform biorobotic arm: A system for the study of spinal circuits, Ann. Biomed. Eng. **23**, 399–408 (1995)

21.28 D.G. Caldwell, G.A. Medrano-Cerda, M. Goodwin: Control of pneumatic muscle actuators, IEEE Control Syst. Mag. **15**(1), 40–48 (1995)

21.29 T. Hesselroth, K. Sarkar, P.P. van der Smagt, K. Schulten: Neural network control of a pneumatic robot arm, IEEE Trans. Syst. Man Cybern. **24**(1), 28–38 (1994)

21.30 A. Bicchi, S.L. Rizzini, G. Tonietti: Compliant design for intrinsic safety: General issues and preliminary design, Proc. IEEE Int. Symp. Intell. Robots Syst. (IROS), Maui (2001) pp. 1864–1869

21.31 F. Daerden, D. Lefeber: The concept and design of pleated pneumatic artificial muscles, Int. J. Fluid Power **2**(3), 41–50 (2001)

21.32 B. Verrelst, R. Van Ham, B. Vanderborght, D. Lefeber, F. Daerden, M. Van Damme: Second generation pleated pneumatic artificial muscle and its robotic applications, Adv. Robotics **20**(7), 783–805 (2006)

21.33 I.M. Gottlieb: *Practical Electric Motor Handbook* (Butterworth-Heinemann, Oxford 1997)

21.34 J.F. Gieras, M. Wing: *Permanent Magnet Motor Technology: Design and Applications* (CRC, Boca Raton 2002)

21.35 M. Gad-el Hak: *The MEMS Handbook: MEMS, Design and Fabrication* (CRC, Boca Raton 2006)

21.36 K. Uchino: *Piezoelectric Actuators and Ultrasonic Motors* (Kluwer, Boston 1997)

21.37 S.-C. Huang, W.-L. Chen: Design of topologically optimal micrograpper, IEEE Int. Conf. Syst. Man Cybern. (2008)

21.38 J. Ueda, T. Secord, H. Asada: Large effective-strain piezoelectric actuators using nested cellular architecture with exponential strain amplification mechanisms, IEEE/ASME Trans. Mechatron. **15**, 770–782 (2010)

21.39 J. Schultz, J. Ueda: Two-port network models for compliant rhomboidal strain amplifiers, IEEE Trans. Robotics **29**(1), 42–54 (2013)

21.40 J. Schultz, J. Ueda: Nested piezoelectric cellular actuators for a biologically inspired camera positioning mechanism, IEEE Trans. Robotics **29**(5), 1–14 (2013)

21.41 J. Schultz, J. Ueda: Experimental verification of discrete switching vibration suppression, IEEE/ASME Trans. Mechatron. **17**(2), 298–308 (2012)

21.42 G. Engdahl: *Handbook of Giant Magnetostrictive Materials* (Academic, San Diego 2000)

21.43 D.C. Lagoudas: *Shape Memory Alloys: Modeling and Engineering Applications* (Springer, New York 2008)

21.44 E. Torres-Jara, K. Gilpin, J. Karges, R.J. Wood, D. Rus: Composable flexible small actuators built from thin shape memory alloy sheets, IEEE Robotics Autom. Mag. **17**(4), 78–87 (2010)

21.45 L. Jinsong: *Shape-Memory Polymers and Multifunctional Composites* (CRC, Boca Raton 2009)

21.46 F. Carpi, E. Smela: *Biomedical Applications of Electroactive Polymer Actuators* (Wiley, Chichester 2009)

21.47 P. Brochu, Q. Pei: Advances in dielectric elastomers for actuators and artificial muscles, Macromol. Rapid Commun. **31**(1), 10–36 (2010)

21.48 H.S. Nalwa: *Ferroelectric Polymers: Chemistry, Physics, and Applications* (CRC, Boca Raton 1995)

21.49 F. Carpi, D. De Rossi, R. Kornbluh, R. Pelrine, P. Sommer-Larsen: *Dielectric Elastomers as Electromechanical Transducers: Fundamentals, Materials, Devices, Models and Applications of an Emerging Electroactive Polymer Technology* (Elsevier, Amsterdan 2008)

21.50 G. Fernandez: Liquid-crystal polymers: Exotic actuators, Nat. Mater. **12**(1), 12–14 (2013)

21.51 R.H. Baughman, C. Cui, A.A. Zakhidov, Z. Iqbal, J.N. Barisci, G.M. Spinks, G.G. Wallace, A. Mazzoldi, D.D. Rossi, A.G. Rinzler, O. Jaschinski, S. Roth, M. Kertesz: Carbon nanotube actuators, Science **284**(5418), 1340–1344 (1999)

21.52 A.E. Aliev, J. Oh, M.E. Kozlov, A.A. Kuznetsov, S. Fang, A.F. Fonseca, R. Ovalle, M.D. Lima, M.H. Haque, Y.N. Gartstein, M. Zhang, A.A. Zakhidov, R.H. Baughman: Giant-stroke, superelastic carbon nanotube aerogel muscles, Science **323**(5921), 1575–1578 (2009)

21.53 J.D.W. Madden: Stiffer than steel, Science **323**(5921), 1571–1572 (2009)

21.54 E. Guglielmino, C.W. Stammers, K.A. Edge, T. Sireteanu, D. Stancioiu: Damp-by-wire: Magnetorheological vs friction dampers, Proc. 16th IFAC World Cong. (2005) pp. 340–345

21.55 M. Laffranchi, N.G. Tsagarakis, D.G. Caldwell: A variable physical damping actuator (VPDA) for compliant robotic joints, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (2010) pp. 1668–1674

21.56 J.C. Dixon: *The Shock Absorber Handbook*, Senior Lect. Eng. Mech. (Wiley, Chichester 2007)

21.57 J. Li, D. Jin, X. Zhang, J. Zhang, W.A. Gruver: An electrorheological fluid damper for robots, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (1995) pp. 2631–2636

21.58 C.-M. Chew, G.-S. Hong, W. Zhou: Series damper actuator: A novel force control actuator, Proc. 4th IEEE/RAS Int. Conf. Humanoid Robots (2004) pp. 533–546

21.59 B. Ebrahimi, M.B. Khamesee, F. Golnaraghi: A novel eddy current damper: Theory and experiment, J. Phys. D Appl. Phys. **42**(7), 075001 (2009)

21.60 H.A. Sodano, J.-S. Bae, D. Inman, K. Belvin: Improved concept and model of eddy current damper, J. Vib. Acoust. **128**(3), 294–302 (2006)

Avidbots Ex1016
Page 567 of 2154

21.61 G.A. Pratt, M.M. Williamson: Series elastic actuators, Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. Hum. Robot Interact. Coop. Robots, Vol. 1 (1995) pp. 399–406

21.62 M.L. Latash: Motor synergies and the equilibrium-point, Mot. Control **14**(3), 294–322 (2010)

21.63 T. Morita, S. Sugano: Development and evaluation of seven-d.o.f. mia arm, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (1997) pp. 462–467

21.64 J.W. Hurst, J. Chestnutt, A. Rizzi: An actuator with physically variable stiffness for highly dynamic legged locomotion, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (2004) pp. 4662–4667

21.65 R. Van Ham, B. Vanderborght, M. Van Damme, B. Verrelst, D. Lefeber: MACCEPA, the mechanically adjustable compliance and controllable equilibrium position actuator: Design and implementation in a biped robot, Robotics Auton. Syst. **55**(10), 761–768 (2007)

21.66 S. Wolf, G. Hirzinger: A new variable stiffness design: Matching requirements of the next robot generation, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (2008) pp. 1741–1746

21.67 N. Hogan: Adaptive control of mechanical impedance by coactivation of antagonist muscles, IEEE Trans. Autom. Control **AC-29**(8), 681–690 (1984)

21.68 S.A. Migliore, E.A. Brown, S.P. DeWeerth: Biologically inspired joint stiffness control, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (2005) pp. 4519–4524

21.69 G. Tonietti, R. Schiavi, A. Bicchi: Design and control of a variable stiffness actuator for safe and fast physical human/robot interaction, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (2005) pp. 528–533

21.70 M.G. Catalano, G. Grioli, M. Garabini, F. Bonomo, M. Mancinit, N. Tsagarakis, A. Bicchi: Vsa-cubebot: A modular variable stiffness platform for multiple degrees of freedom robots, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (2011) pp. 5090–5095

21.71 A. Bicchi, G. Tonietti: Fast and *soft-arm* tactics, IEEE Robotics Autom. Mag. **11**(2), 22–33 (2004)

21.72 A. Albu-Schäffer, S. Wolf, O. Eiberger, S. Haddadin, F. Petit, M. Chalon: Dynamic modelling and control of variable stiffness actuators, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (2010) pp. 2155–2162

21.73 A.H.C. Gosline, V. Hayward: Eddy current brakes for haptic interfaces: Design, identification, and control, IEEE/ASME Trans, Mechatron. **13**(6), 669–677 (2008)

21.74 M. Catalano, G. Grioli, M. Garabini, F.W. Belo, A. di Basco, N. Tsagarakis, A. Bicchi: A variable damping module for variable impedance actuation, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (2012) pp. 2666–2672

21.75 VIACTORS (Deutsches Zentrum für Luft- und Raumfahrt, Università di Pisa, Universiteit Twente, Imperial College London, Instituto Italiano di Tecnologia, Vrije Universiteit Brussel): Variable Impedance Actuators data sheets, http://www.viactors.org/VSA%20data%20sheets.htm (2011)

21.76 G. Grioli, S. Wolf, O. Eiberger, W. Friedl, M. Grebenstein, H. Höppner, E. Burdet, D. Caldwell, R. Carloni, M. Catalano, D. Lefeber, S. Stramigioli, N. Tsagaratkis, M. Van Damme, R. Van Ham, B. Vanderborght, L.C. Visser, A. Bicchi, A. Albu-Schaeffer: Variable stiffness actuators: The user's point of view, Int. J. Robotics Res. **34**(6), 727–743 (2015)

21.77 S. Wolf, G. Grioli, W. Friedl, M. Grebenstein, H. Hoeppner, E. Burdet, D. Caldwell, A. Bicchi, S. Stramigioli, B. Vanderborght: Variable stiffness actuators: Review on design and components, IEEE/ASME Trans. Mechatron. **PP**(99), 1 (2015)

21.78 M. Krasnosel'skii, A. Pokrovskii: *Systems with Hysteresis* (Springer, Berlin, Heidelberg 1989)

21.79 A.G. Feldman: Functional tuning of the nervous system with control of movement or maintenance of a steady posture. II: Controllable parameters of the muscle, Biophysics **11**, 565–578 (1966)

21.80 A.C. Antoulas: *Approximation of Large-Scale Dynamical Systems* (SIAM, Philadelphia 2005)

21.81 I. Fantoni, R. Lozano, M.W. Spong: Energy based control of the pendubot, IEEE Trans. Autom. Control **45**(4), 725–729 (2000)

21.82 A. Jafari, N. Tsagarakis, D. Caldwell: A novel intrinsically energy efficient development of a novel actuator with adjustable stiffness (awas), IEEE Trans. Mechatron. **18**(1), 355–365 (2013)

21.83 A. De Luca: Decoupling and feedback linearization of robots with mixed rigid/elastic joints, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (1996) pp. 816–821

21.84 A. De Luca, P. Lucibello: A general algorithm for dynamic feedback linearization of robots with elastic joints, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (1998) pp. 504–510

21.85 M. Spong: Modeling and control of elastic joint robots, ASME J. Dyn. Syst. Meas. Control **109**, 310–319 (1987)

21.86 A. Albu-Schffäer, S. Wolf, O. Eiberger, S. Haddadin, F. Petit, M. Chalon: Dynamic modeling and control of variable stiffness actuators, Proc. IEEE Conf. Robotics Autom. ICRA (2010) pp. 2155–2162

21.87 M. Grebenstein, A. Albu-Schäffer, T. Bahls, M. Chalon, O. Eiberger, W. Friedl, R. Gruber, S. Haddadin, U. Hagn, R. Haslinger, H. Hoppner, S. Jörg, M. Nickl, A. Nothhelfer, F. Petit, J. Reill, N. Seitz, T. Wimböck, S. Wolf, T. Wusthoff, G. Hirzinger: The DLR hand arm system, Proc. IEEE Int. Conf. Robotics Autom. (ICRA), Shanghai (2011) pp. 3175–3182

21.88 H.P. Olesen, R.B. Randall: *A Guide to Mechanical Impedance and Structural Response Techniques*, Tech. Rep. (Bruel and Kjaer, Noerum 1977)

21.89 H. Gomi, M. Kawato: Human arm stiffness and equilibrium-point trajectory during multi-joint movement, Biol. Cybern. **76**(3), 163–171 (1997)

21.90 N. Hogan: Impedance Control: An approach to Manipulation – Parts I–III, J. Dyn. Syst. Meas. Control **107**, 1–24 (1985)

21.91 F.A. Mussa-Ivaldi, N. Hogan, E. Bizzi: Neural, mechanical, and geometric factors subserving

Avidbots Ex1016
Page 568 of 2154

arm posture in humans, J. Neurosci. **5**(10), 2732 (1985)
21.92 H. Gomi, R. Osu: Task-dependent viscoelasticity of human multijoint arm and its spatial characteristics for interaction with environments, J. Neurosci. **18**(21), 8965 (1998)
21.93 E. Burdet, R. Osu, D.W. Franklin, T.E. Milner, M. Kawato: Measuring stiffness during arm movements in various dynamic environments, Proc. 1999 ASME Annu. Symp. Haptic Interfac. Virtual Environ. Teleoper. Syst. (1999) pp. 421–428
21.94 E. Burdet, R. Osu, D.W. Franklin, T. Yoshioka, T.E. Milner, M. Kawato: A method for measuring endpoint stiffness during multi-joint arm movements, J. Biomech. **33**(12), 1705–1709 (2000)
21.95 D.W. Franklin, E. Burdet, R. Osu, M. Kawato, T.E. Milner: Functional significance of stiffness in adaptation of multijoint arm movements to stable and unstable dynamics, Exp. Brain Res. **151**(2), 145–157 (2003)
21.96 K.P. Tee, E. Burdet, C.M. Chew, T.E. Milner: A model of force and impedance in human arm movements, Biol. Cybern. **90**(5), 368–375 (2004)
21.97 E.J. Perreault, R.F. Kirsch, P.E. Crago: Multijoint dynamics and postural stability of the human arm, Exp. Brain Res. **157**(4), 507–517 (2004)
21.98 R.D. Trumbower, M.A. Krutky, B.S. Yang, E.J. Perreault: Use of self-selected postures to regulate multi-joint stiffness during unconstrained tasks, PLoS ONE **4**(5), e5411 (2009)
21.99 K. Hashimoto, T. Kureha, Y. Nishimura, K. Okumura, S. Muraoka: Measurement of mechanical impedance using quartz resonator force sensor during the process of grasping, SICE Annu. Conf., Vol. 1 (2004) pp. 722–726
21.100 A. Serio, G. Grioli, I. Sardellitti, N.G. Tsagarakis, A. Bicchi: A decoupled impedance observer for a variable stiffness robot, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (2011) pp. 5548–5553
21.101 F. Flacco, A. De Luca, I. Sardellitti, N. Tsagarakis: On-line estimation of variable stiffness in flexible robot joints, Int. J. Robotics Res. **31**(13), 1556–1577 (2012)
21.102 T. Ménard, G. Grioli, A. Bicchi: A real time observer for an agonist-antagonist variable stiffness actuator, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (2013)
21.103 T. Ménard, G. Grioli, A. Bicchi: A stiffness estimator for agonistic-antagonistic variable-stiffness-actuator devices, IEEE Trans. Robotics **30**(5), 1269–1278 (2014)
21.104 G. Grioli, A. Bicchi: A non-invasive real-time method for measuring variable stiffness, Proc. 10th Int. Conf. Robotics Sci. Syst., Zaragoza (2010)
21.105 J.K. Salisbury: Active stiffness control of a manipulator in cartesian coordinates, Proc. 19th IEEE Conf. Decis. Control (1980) pp. 83–88
21.106 C. Ott: *Cartesian Impedance Control of Redundant and Flexible-Joint Robots*, Springer Tracts. Adv. Robotics (Springer, Berlin, Heidelburg 2008)
21.107 N. Hogan: Mechanical impedance of single- and multi-articular systems. In: *Multiple Muscle Systems*, ed. by J.M. Winters, S.L.-Y. Woo (Springer, New York 1990) pp. 149–164
21.108 A. Albu-Schäffer, M. Fischer, G. Schreiber, F. Schoeppe, G. Hirzinger: Soft robotics: What cartesian stiffness can we obtain with passively compliant, uncoupled joints?, Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS) (2004)
21.109 F. Petit, A. Albu-Schäffer: Cartesian impedance control for a variable stiffness robot arm, Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (2011)
21.110 G. Palli, C. Melchiorri, A. De Luca: On the feedback linearization of robots with variable joint stiffness, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (2008)
21.111 A. De Luca, F. Flacco, A. Bicchi, R. Schiavi: Nonlinear decoupled motion-stiffness control and collision detection/reaction for the VSA-II variable stiffness device, Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS) (2009)
21.112 G. Palli, C. Melchiorri: Output-based control of robots with variable stiffness actuation, J. Robotics (2011) doi:dx.doi.org/10.1155/2011/735407
21.113 A. Albu-Schäffer, C. Ott, F. Petit: Energy shaping control for a class of underactuated Euler-Lagrange systems, Proc. IFAC Symp. Robot Control (2012)
21.114 I. Sardellitti, G. Medrano-Cerda, N.G. Tsagarakis, A. Jafari, D.G. Caldwell: A position and stiffness control strategy for variable stiffness actuators, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (2012)
21.115 F. Petit, A. Albu-Schäffer: State feedback damping control for a multi dof variable stiffness robot arm, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (2011)
21.116 B. Vanderborght: Dynamic Stabilisation of the Biped Lucy Powered by Actuators with Controllable Stiffness, Ph.D. Thesis (Vrije Universiteit Brussel, Brussel 2007)
21.117 M. Uemura, S. Kawamura: Resonance-based motion control method for mulit-joint robot through combining stiffness adaptation and iterative learning control, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (2009)
21.118 L.C. Visser, S. Stramigioli, A. Bicchi: Embodying desired behavior in variable stiffness actuators, Proc. IFAC Congr. Int. Fed. Autom. Control (2011)
21.119 D. Lakatos, F. Petit, A. Albu-Schäffer: Nonlinear oscillations for cyclic movements in variable impedance actuated robotic arms, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (2013) pp. 503–513
21.120 D. Lakatos, G. Garofalo, F. Petit, C. Ott, A. Albu-Schäffer: Modal limit cycle control for variable stiffness actuated robots, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (2013)
21.121 A. Velasco, G.M. Gasparri, M. Garabini, L. Malagia, P. Salaris, A. Bicchi: Soft-actuators in cyclic motion: Analytical optimization of stiffness and preload, Proc. IEEE/RAS Int. Conf. Humanoid Robots, Atlanta (2013)
21.122 G. Garofalo, C. Ott, A. Albu-Schäffer: Orbital stabilization of mechanical systems through semidef-

Avidbots Ex1016
Page 569 of 2154

inite Lyapunov functions, Proc. Am. Control Conf. (ACC) (2013)
21.123 D. Lakatos, F. Petit, A. Albu-Schäffer: Nonlinear oscillations for cyclic movements in human and robotic arms, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (2013) pp. 508–511
21.124 M. Papageorgiou: *Optimierung: Statische, Dynamische, Stochastische Verfahren* (Oldenbourg, Munich 1996)
21.125 A. Bicchi, G. Tonietti: Fast and soft arm tactics: Dealing with the safety-performance trade-off in robot arms design and control, IEEE Robotics Autom. Mag. **11**(2), 22–33 (2004)
21.126 S. Haddadin, M.C. Özparpucu, A.A. Schäffer: Optimal control for maximizing potential energy in variable stiffness joints, Proc. 51st IEEE Conf. Decis. Control (CDC), Maui (2012)
21.127 S. Haddadin, M. C. Özparpucu, F. Huber, N. Mansfeld, A. Albu-Schäffer: Exploiting the natural dynamics of elastic joints for peak velocity, Int. J. Robotics Res. (2012)
21.128 M. Garabini, A. Passaglia, F. Belo, P. Salaris, A. Bicchi: Optimality principles in stiffness control: The vsa kick, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (2012) pp. 3341–3346
21.129 R. Incaini, L. Sestini, M. Garabini, M.G. Catalano, G. Grioli, A. Bicchi: Optimal control and design guidelines for soft jumping robots: Series elastic actuation and parallel elastic actuation in comparison, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (2013) pp. 2477–2484
21.130 D. Garg, M.A. Patterson, W.W. Hager, A.V. Rao, D.A. Benson, G.T. Huntington: A unified framework for the numerical solution of optimal control problems using pseudospectral methods, Automatica **46**(11), 1843–1851 (2010)
21.131 W. Li, E. Todorov: Iterative linearization methods for approximately optimal control and estimation of non-linear stochastic system, Int. J. Control **80**(9), 1439–1453 (2007)
21.132 W. Li, E. Todorov: Iterative linear quadratic regulator design for nonlinear biological movement systems, Proc. Int. Conf. Inform. Control, Autom. Robotics (2004) pp. 222–229
21.133 S. Haddadin, F. Huber, A. Albu-Schaffer: Optimal control for exploiting the natural dynamics of variable stiffness robots, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (2012) pp. 3347–3354
21.134 D. Benson: A Gauss Pseudospectral Transcription for Optimal Control, Ph.D. Thesis (MIT, Cambridge 2005)
21.135 IEEE Robotics and Automation Society Technical Committee on Soft Robotics: http://www.ieee-ras.org/soft-robotics
21.136 Natural Machine Motion Initiative: http://naturalmachinemotion.com

Avidbots Ex1016
Page 570 of 2154

Multimedia Contents

22. Modular Robots

I-Ming Chen, Mark Yim

This chapter presents a discussion of modular robots from both an industrial and a research point of view. The chapter is divided into four sections, one focusing on existing reconfigurable modular manipulators typically in an industry setting (Sect. 22.2) and another focusing on self-reconfigurable modular robots typically in a research setting (Sect. 22.4). Both sections are sandwiched between the introduction and conclusion sections.

This chapter is focused on design issues. Rather than a survey of existing systems, it presents some of the existing systems in the context of a discussion of the issues and elements in industrial modular robotics and modular robotics research. The reader is encouraged to look at the references for further discussion on any of the presented topics.

22.1 **Concepts and Definitions** 531
22.1.1 Concept of Modularity 532
22.1.2 Definition and Classification of Modular Robots 532
22.2 **Reconfigurable Modular Manipulators** 533
22.2.1 Background of Modular Manipulator Systems 533
22.2.2 Module Design Issues 534
22.2.3 Interface Design Issues 534
22.2.4 Modeling of Modular Manipulators 534
22.2.5 Configuration Optimization 535
22.3 **Self-Reconfigurable Modular Robots** 535
22.3.1 Types of Self-Reconfigurable Modular Robots 536
22.3.2 System and Module Design Issues 536
22.3.3 Interface Design Issues 537
22.4 **Conclusion and Further Reading** 539
Video-References 540
References 540

Modularity in design engineering refers to a compartmentalization of elements. Most often modularity in complex systems occurs as a result of taking a complex system and dividing it into pieces in order to better understand the simpler elements and parallelize the design efforts. Modularity also facilitates the replacing of elements either for repair or upgrading new functionality. The alternative to a modular approach is an integrated approach where systems are designed as a whole. While integrated approaches tend not to be as easy to repair, upgrade or reconfigure, they do have fewer constraints on element design and therefore can be made more optimal. Integrated approaches can focus on lowering cost or having higher performance. In mechanical devices, the choice between modular or integrated architectures can have a large impact on the range of application as well as cost or performance [22.1].

22.1 Concepts and Definitions

For example, a hand drill with modular attachments can expand itsrange of functionality from drilling holes to screwing bolts or buffing surfaces. For robotics, the same impact applies. However, robots have an inher-

ent complexity that lends itself to modularity,such as actuator modules, sensor modules, and sometimes computation modules. In the following sections, definitions will be given for different kinds of modularity frequently seen in robotics.

22.1.1 Concept of Modularity

The product design literature can be looked at as encompassing robotics – which can be considered as an industrial or research product. The architecture of product modularity can be categorized into three subtypes: slot, bus, and sectional modularity [22.1]:

- *Slot architecture*: Each of the interfaces between components is of a different type from the others, so that the various components in the product cannot be interchanged.
- *Bus architecture*: There is a common bus to which the other physical components connect via the same type of interface.
- *Sectional architecture*: All interfaces are of the same type and there is no single element to which all the other components attach, i. e., there is no base component. The assembly is built up by connecting the components to each other via identical interfaces (Fig. 22.1).

Such classifications provide a good definition of modular robots as follows:

- If a complex robotic system adopts a slot- and bus-modularity design approach for its internal structure and architecture, not the external configuration, it can be called a *modularly designed robotic system* benefiting from design parallelization. Such a robot may have a unified and integrated configuration that cannot be changed from outside.
- If a robot adopts a bus- and sectional-modularity design approach for both internal structure and external configuration, it can be called a *modular robot*. The users can reconfigure the compartmentalization and interchange functional modules with some level of effort.

22.1.2 Definition and Classification of Modular Robots

Any system can be reconfigured by destructing and reconstructing it, for example in the worst case, using a blowtorch and milling machine. The key element we must define is the level of effort required to reconfigure. We propose here three levels from the lowest to the highest level of effort of the user:

1. The system reconfigures itself. It is *self-reconfigurable*.
2. The system is reconfigured by a lay user with or without special tools typically in matter of seconds to minutes.
3. The system is reconfigured only by an expert with specialized tools.

This chapter will focus on modular robots that give rise to *plug-and-play* reconfiguration of the system for task and function changes, levels 1 and 2. Level 1 systems are self-reconfigurable systems with sectional modularity. Level 2 systems in this chapter focuses on reconfigurable modular manipulators with a finite set of modules of different functions.

The modular manipulator type of robotic systems are natural evolution of industrial robot manipulators that consist of a number of specific functional modules, such as actuator modules, link modules, and end-effector modules. Subsequently, robots with the serial and branching topology, such as humanoid robots, legged robots, mobile manipulators adopt a similar approach for modularity as these functional modules form the basis of a robotic system.

The self-reconfigurable modular robots grew out of the concept of self-evolution and self-configuration of biological cells with identical units. Such a robot normally consists of a large number of a small set of types of mechatronic units that possess actuation, connection, communication, and computing capability that can be assembled together in arbitrary forms and also reconfigure itself.

Although the two types of modular robots originated from different fundamental concepts, the goals to provide a large number of possible robot configurations for different tasks with the same set of basic robot modules are the same.

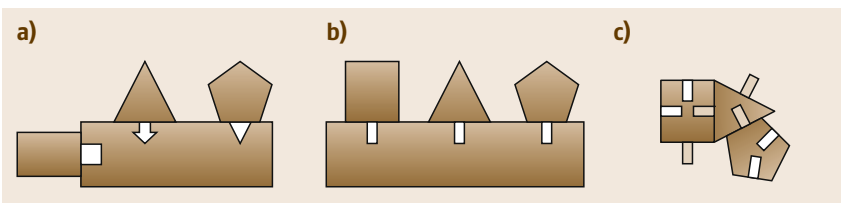


Fig.22.1a–c Modular architecture types: **(a)** slot architecture **(b)** bus architecture **(c)** sectional architecture

Avidbots Ex1016
Page 572 of 2154

22.2 Reconfigurable Modular Manipulators

Reconfigurable modular manipulators are robot arms that have elements that can be rearranged.

22.2.1 Background of Modular Manipulator Systems

The simplest form of modular manipulator comes in the form of automatic tool changers also called quick-change end-effectors. These are optional tools that can be attached to the end of a robotic arm. Automatic tool changers are standard equipment for many CNC (computer numerically controlled) milling machines and lathes since the 1960s. They enable the machines to drill holes of different sizes, or cut different shapes. Although CNC machines are not often considered industrial robots, they share the same elements (actuators, sensors, and computation) and their function is more limited. Most industrial robot arms can be equipped with automatic changing end-effectors by adding a wrist that has the compatible interface for a variety of grippers and end-effectors. These devices are available commercially from Schunk (Germany), ATI industrial automation (USA), Destaco, Amatrol, RE2 (USA), RAD, and others.

In the modularization of industrial robots, the granularity of the components is usually based on their basic functions, i. e., motion actuation and tooling. Thus, the design of modules is highly differentiated into actuator modules, passive joint modules, and tooling modules, etc. Several prototype modular robotic systems have been built and demonstrated, including the reconfigurable modular manipulator system (RMMS) [22.2], several generations of the cellular robotic system (CEBOT) [22.3], and modular manipulator systems developed by University of Toronto [22.4], University of Stuttgart [22.5], University of Texas at Austin [22.6], and Toshiba Corp. [22.7].

Basically, these systems have serial-type (or open-chain) geometry with large working envelopes. These serial-type modular robots are well suited for assembly, trajectory tracking, welding, and hazardous material handling. Parallel modular robots have also been developed for light-machining tasks [22.8]. As indicated in [22.8], modular design can reduce the development cycle of the parallel robots significantly. Furthermore, it allows a trial-and-error approach for the construction of parallel robots that is impossible with the integrated design approach.

With globalization of world manufacturing, the concept of modular manipulators has quickly gained industrial attention. A full-scale reconfigurable robotic system workcell consisting of three modular robots with a total 15 axes was successfully showcased in 1999 [22.9] (Fig. 22.2). The modular robot workcell has a serial type 6-degrees-of-freedom (DOF) robot for the pick-and-place action of the work piece, a 2-DOF robot for work piece transfer, and a 6-DOF parallel robot for milling operations on the work piece. All the robots are built from the same set of modular components, including actuator modules, link modules, and tool modules.

The German company Amtec, later acquired by Schunk, developed the first commercially available modular manipulator system called PowerCube. Schunk has subsequently developed its industrial manipulators and automation systems based on PowerCube with some success [22.10]. Today there are many robotic systems with a wide spectrum of applications that are built around modular robot components.

The modular robot concept also proliferated in the hobby and educational robot sectors around the year 2000 by the introduction of well-packaged self-contained servo motor modules into inexpensive robotic devices, such as Robotis (Korea) and Kondo (Japan), as well as Lego (Denmark) and other toy companies making educational robots.

Mobile robots with legs, wheels, and tracks also belong to this class of modularity where they are configured for different task requirements such as those needed for disaster relief, rescue, and surveillance purposes. Two tracked modular mobile robots designed with multiple track segments [22.11] and reconfigurable tracks allowing serial and parallel connections [22.12] have been demonstrated. The work in [22.13] contains an in-depth review of the development in modular mobile robots.

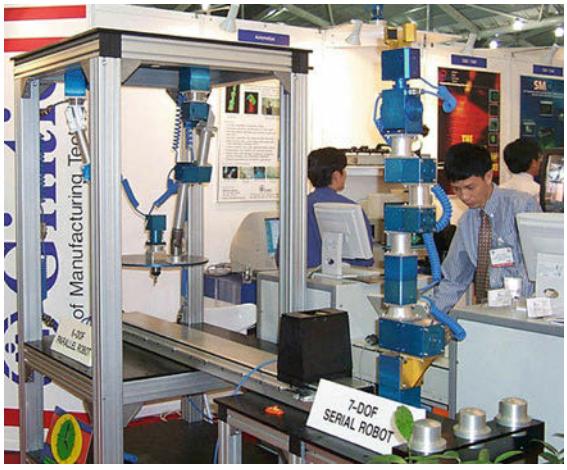


Fig. 22.2 15-axis Reconfigurable Robotic Workcell (after [22.9])

Avidbots Ex1016
Page 573 of 2154

22.2.2 Module Design Issues

A modular robot consists of the two main features found in a modular product: 1) a one-to-one mapping from functional elements to the physical components of the product and 2) decoupled interfaces between the components of different modules [22.1]. For modular manipulators, the essential components are the base, positioning, and orienting mechanisms composed by actuator modules and link modules of different dimensions and geometry, and the end-effector module. For legged and wheeled mobile robotic systems, the motion generation mechanism modules are essential.

The actuator modules normally use DC or AC motors as a 1-DOF *rotate* or *pivot* joint module typically with compact high reduction ratio transmission mechanisms [22.2, 4, 6, 7, 9]. Some modular systems also adopt 1-DOF linear modules for larger workspace envelope [22.4, 9] and 2-DOF joint modules for compact dexterous motions [22.10]. The actuator module shown in Fig. 22.3 has two independent linear and rotary motion capabilities suitable for compact assembly tasks [22.9]. The actuator modules are typically designed with similar geometry but with different dimensions and power ratings for different application requirements.

The link modules connecting units in between the actuators function as reachable workspace extenders. Some systems adopt a standard fixed-dimension connection module [22.3, 4, 6] and some use variable dimension modules that can be customized to satisfy arbitrary design constraints [22.9]. In some systems [22.2], the link module becomes part of the actuator module so that the module acts as an actuator as well as the connecting structure.

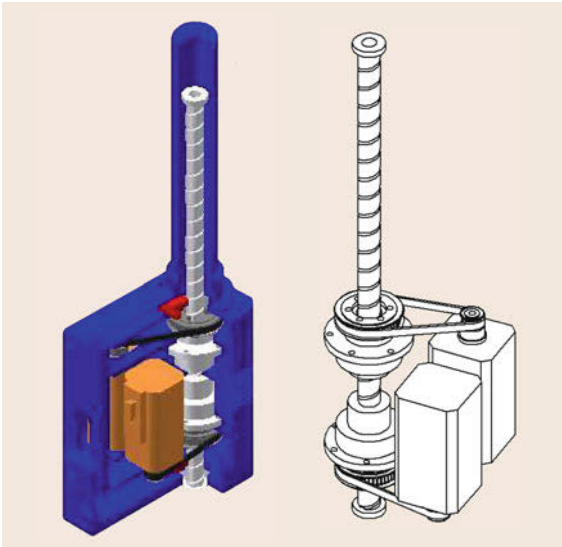


Fig. 22.3 A 2-DOF translate-turn module with ball-screw and ball-spline mechanism

22.2.3 Interface Design Issues

The mechanical connecting interface between modules in a modular manipulator needs to satisfy three basic requirements:

1. Stiffness
2. Fast reconfiguration
3. Interchangeability.

Thus, the design of mechanical connections or docking mechanisms is a critical issue. In a fully or semisupervised robotic system, like some modular manipulator systems [22.2–9], the connecting mechanism is designed to be manually operated for reliability and safety reasons. In a fully autonomous system, the connecting mechanism needs to be designed typically with an extra actuator and locking mechanism for carrying out the connection automatically. This is the case for most of the self-reconfigurable modular robots.

In order to meet the requirement of interchangeability, the electronic and communication interface for the modular system normally adopts common communication network architecture with plug-and-play capability similar to local area network (LAN). There are a number of existing industrial standard network protocols for real-time robot control suitable for such applications, like CAN-bus, RS485, and IEEE 1394. The progressive development of industrial automation protocols will facilitate the implementation of modular robot communications. While many of the early systems use wired multidrop bus architectures for communications, multirobot systems have used fast local message forwarding [22.14] and wireless networking.

22.2.4 Modeling of Modular Manipulators

Challenges to model the modular manipulator systems come from the lack of uniform formalisms of the unfixed robot configuration and geometry and the errors accumulated from assembly and dis-assembly of the modules. Hence, the first effort in modular robot modeling was the introduction of a graph-based technique with additional module assembly information for the representation of a modular robot configuration [22.15]. This work introduced a modular robot representation scheme, termed assembly incidence matrix (AIM) for distinct modular robot configurations. There are several subsequent extensions and variations of AIM for broader categories of modular robots including modular mobile manipulators [22.16–18].

Avidbots Ex1016
Page 574 of 2154

Once the modular robot configuration can be distinctly defined with the type of modules, the connection sequence, module orientation, kinematic, and dynamic models of the robot can be obtained through an automatic generation algorithm [22.19]. Kinematic model generation can be achieved through the conventional Denavit–Hartenberg (DH) parameterization [22.2–4, 17] or the coordinate-free local product-of-exponential (POE) approach [22.8, 9, 19]. However, the DH method does not provide a clear distinction between the arranging sequence of modules in a robot chain, and it is an initial position-dependent representation. The local POE formulation of the kinematics and dynamics based on the theory of Lie groups and Lie algebras for rigid motion in $se(3)$ and $SO(3)$ can avoid this problem. Furthermore, the POE representation can avoid the singularity conditions that frequently occur in the kinematic calibration formulated by the DH method [22.20]. Thus, POE representations provide a uniform and well-behaved method for handling the inverse kinematics of both calibrated and un-calibrated robot systems. In local POE modeling, the joint axes are described in the local module (body) coordinate systems, it is progressive in constructing the kinematic models, so it conveniently resembles the assembling action of the physical modular robot components.

The machining tolerance, compliance, and wear of the connecting mechanism due to frequent module reconfiguration may introduce errors in positioning the end effector. Hence, kinematic calibration is a must for modular robots. In the POE calibration model, the robot errors are assumed to be in the initial positions of the consecutive modules because the local POE model is a zero reference method. Based on linear superposition and differential transformation, a 6-parameter error model can be established for serial-type robots [22.19]. This model can be obtained through the automatic generation process. An iterative least-square algorithm employed to find the error parameters to be corrected. The corrected kinematic model is then updated in the robot controller for operation. The simulation and experiment have shown that the proposed method can improve the position accuracy up to two orders of magnitude, or to the nominal repeatability of the robot after calibration with measurement noise. A typical 6-DOF articulate-type modular robot can reach a position accuracy of 0.1 mm compared to an accuracy of 1 mm before the calibration [22.20]

A formulation of the dynamic model of modular manipulators starts from a recursive Newton–Euler algorithm [22.21, 22]. The generalized velocity, acceleration, and forces can be expressed in terms of linear operations on $se(3)$ [22.23]. Based on the relationship between the recursive formulation and the closed-form Lagrangian formulation for serial-robot dynamics discussed in [22.24, 25], the AIM can assist in the construction of the closed-form equation of motion of a modular robot in any generic topology with redundant and non-redundant configurations [22.19].

22.2.5 Configuration Optimization

Due to the modular design, the modular manipulator can be optimal at the component level, but may not obtain optimal performance at the system level. Task-driven robot configuration optimization becomes necessary to establish locally optimal performance for the overall robotic system. Typically, the problem of robot configuration optimization can be stated as finding an assembly of robot modules that can achieve a certain task requirement based on an inventory of modules. The configuration of a modular robot can be treated as a compound entity with finite number of constituents. Finding the most suitable task-oriented robot configuration then becomes a discrete design optimization problem using a task performance related objective function. Discrete optimization techniques, such as genetic algorithms (GAs), the simulated annealing (SA) method, and other artificial intelligence techniques have been employed to find solutions [22.26–28].

The criteria used in selecting the optimal configuration depend largely on the task requirements, which mostly describe the necessary robot trajectories or key postures. *Yang* and *Chen* proposed a reduced DOF approach to minimize the total number of actuator modules employed in a serial-type modular robot for a given task [22.29]. With fewer modules, the robot can carry more payloads instead of carrying distal modules. Furthermore, the robot can be operated at higher speed with better dynamic response.

22.3 Self-Reconfigurable Modular Robots

Self-configurable systems can rearrange their own topology. An example is shown in VIDEO 2. There are dozens of research groups who have constructed many versions of self-reconfigurable robots [22.3, 30–46], with many approaches for programming them [22.36, 47–61]. As of 2012, over 800 papers and a book [22.31] have been written.

These systems are characterized by many identical modules that can be rearranged into a variety of shapes and configurations and by being highly scal-

Avidbots Ex1016
Page 575 of 2154

able with simulated systems having hundreds, thousands, or millions of modules. These systems have three promises:

1. Low cost from batch fabrication of repeated modules
2. High robustness from redundancy and the ability to self-repair
3. High versatility from the ability to reconfigure and adapt to changing situations [22.62].

Practically speaking, none of the promises have been proven, though the promise of versatility is getting close. These systems have exhibited a wide variety of locomotion and manipulation including: legged walking with between 2 and 14 legs; riding a tricycle [22.63]; rolling like a tread [22.14]; snake-like locomotion (lateral and rectilinear undulation, concertina, sidewinding) [22.64]; manipulation of large objects with multiple arms/fingers [22.65]; manipulation of small objects; climbing stairs, fences, poles, in pipes; self-reconfiguration between dozens of shapes and many others [22.31]. Figure 22.4 and VIDEO 1 show a self-reconfigurable modular robot SMORES [22.66]. Other self-reconfigurable robots like ATRON [22.40] is shown in VIDEO 5 and M-blocks is shown in VIDEO 3.

22.3.1 Types of Self-Reconfigurable Modular Robots

Self-reconfiguring systems can be classified into three types based on the style of reconfiguration: chain, lattice, and mobile [22.62]. The chain systems reconfigure by using chains of modules that form and break loops [22.36, 62]. They tend to be well suited for work on the environment, as they can form articulated limbs. The lattice systems have modules which have nominal positions sitting on a regular lattice and tend to be better at self-reconfiguration as moving to neighboring lattice positions makes collision checking easy [22.33, 34, 39, 43]. The mobile systems have modules that individually maneuver on terrain and reconfigure by moving on the environment to relocate themselves in a conglomerate [22.38, 67].

Of the systems that have been implemented to date, some that have been shown to be most capable (judging by number of demonstrations) are the hybrid chain-lattice systems: Superbot [22.14], MTRAN III [22.68] and CKbot [22.69]. Recent additions to this group are Johns Hopkins University [22.70], iMobot [22.71], and SMORES [22.66] which are hybrid in all three areas, chain-lattice-mobile systems.

22.3.2 System and Module Design Issues

There is an interesting phenomenon called second system syndrome [22.72], where designers include many features in the second version of a system they design. They often include many more features than the system may need. This is especially problematic in modular reconfigurable systems with repeated modules; any feature added to one module has the possibility of its effect multiplied by n, the number of modules. For example, an increase of d in computational processing power (e.g., microprocessor without interlocked pipeline stages, MIPS) in one module will result in a system increase of dn.

Another similar phenomenon for designers is called *feature creep*, where more and more features are added as the system is being designed. Often this has a cumulative linear effect on cost, but worse, there is an exponential effect on reliability.

The simplest form of robustness analysis assumes independent probabilities that a module may fail during a specified function. If one module succeeds during that function with probability p then the system with $n$identical module has a probability of success of p^n, under the assumption that all modules must be functioning to succeed. Clearly this is problematic for large n.

As systems scale up in number, a key to make things work is to be sure that the system does not depend on every module working perfectly. Indeed, if only X numbers are needed, one would wonder why one would use more than X, especially if the only impact is reduced reliability. Here, one strategy is to ensure that the solution is devised in a way where the performance improves with the number of modules. If modules fail, then the system can gracefully degrade. In systems with tightly coupled actions between modules, this can be difficult

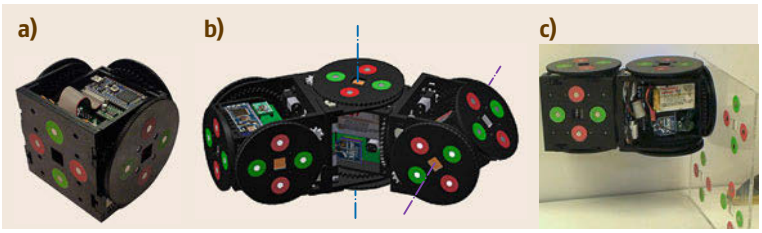


Fig. 22.4 **(a)** One SMORES module with four main actuators and four docking faces. **(b)** Three SMORES modules attached together. **(c)** Two modules moving on a lattice

Avidbots Ex1016
Page 576 of 2154

to implement. In tasks where only binary metrics exist, (success or failure) using the optimal X modules, setting aside the extras may be the best solution.

One of the aspects that researchers find most interesting about self-reconfigurable robots is examining what happens when modules scale up in number (and also typically, scaled down in size). As the numbers increase, the number of shapes/configurations increases and concomitantly, the types of activities that are possible increase as well.

Control and planning get very complicated quickly though. Simply enumerating the number of isomorphic configurations has resulted in a PhD thesis without completely solving the problem [22.15].

While researchers have simulated hundreds, thousands, and even millions of modules, no physical system has been demonstrated with more than a few hundred to date. The largest single system so far is Kilobot [22.73], which has very simple mobile modules that swarm together and actually do not connect rigidly. The modules communicate wirelessly in a one-to-many (broadcast) fashion. For rigidly connected systems the largest number of modules demonstrated in one connected system was a 14-legged 48-module Polybot system [22.74].

Main Actuation

Every module has some form of actuation that enables the modules to move from one position to another or to do some work on the environment. We call it the *main actuator*. By far the most common main actuator is a DC electric motor as it is the lowest cost and easiest to implement.

In chain systems, the main actuator typically drives a revolute joint so that a chain of modules forms an articulated arm. In lattice systems, the main actuator typically moves the module (or a neighboring module) along a constrained 1-DOF path that can be translation or rotation. In mobile reconfiguration, the main actuator enables locomotion, usually through wheels or treads.

The main actuator is typically the largest component in a module and so has been the focus when trying to make modules smaller. To date, the smallest module with onboard actuators, that both attaches and detaches uses shape memory actuated module that was 2 cm [22.75]. An even smaller 1 mm module was created at Carnegie Mellon University [22.76] using electrostatics as an actuator, but it did not attach to other modules.

On the other extreme, the largest module is GHC [22.77], an 8 m^3 helium-filled cube with shape memory actuators on the edges to rotate these floating balloons about edges, attaching electrostatically. DARPA is also sponsoring a project to look at reconfigurable maritime craft constructed in a 20′ ISO container form factor. University of Pennsylvania has demonstrated scale models at 1/12 scale, with main actuators being thrusters to maneuver the parallelepipeds in water [22.78].

22.3.3 Interface Design Issues

The main component of modular systems differentiating it from an integrated system is the interface between modules. When talking about modular systems, the amount of *modularity* can be measured by the number of these interfaces. For self-reconfigurable modular robotic systems in this chapter the number of interfaces in one connected component can be as small as six or as large as millions.

In the most general case, every interface must do two things: 1) attach and 2) detach. When they attach they must do two things: 1) form a physical coupling and 2) allow a flow through the interface for supply power and information (often done electrically).

The interfaces can be *gendered* [22.36], that is two mating interfaces are not identical, one has male features sticking out and the other female features to receive the male features. They can be *ungendered* with no protruding or receding mating features. Or they can be *hermaphroditic* [22.39] where interfaces contain both male and female features. Ungendered and hermaphroditic interfaces can have identical interfaces on both sides. This increases the number of possible arrangements over gendered modules; however, ungendered and hermaphroditic components are typically more complex than gendered interfaces.

Each module can have multiple interfaces. If we consider the number of configurations possible with m interfaces per module assuming identical modules and ignoring any physical constraints such as self-collision, we obtain m^n possible configurations. The number of non-isomorphic configurations is much less, but is very difficult to enumerate in general. However, one special case to consider is two interfaces per module. In this case, topologically, there are only two different configurations, a single chain, and a loop. For this reason, most systems have three or more interfaces.

In addition, each interface can have multiple symmetries that allow different kinematic relationships for the same adjacency relationship. For example many systems have either a 2-way connector (modules may be optionally be rotated 180° to each other) or a 4-way connector (90° rotations.) This p-way symmetry leads to $(pm)^n$ total possible configurations.

Mechanical Interfaces

Mechanical interfaces rigidly attach two modules together with the ability to detach as well. Strategies to

enable this capability include using nuts and bolts, magnetic bonding [22.39], electrostatic bonding [22.77], probe and drogue [22.79], and a variety of physical hook-type mechanisms. Each mechanical interface has three performance metrics: bond strength, acceptance range, and interface precision/stiffness.

Bond strength can be characterized by the amount of force required to separate two modules in its weakest direction [22.80]. The acceptance range is the tolerance to position errors as two docks approach to dock [22.81]. Interface precision/stiffness indicates the amount of position error (deflection) that can occur contributed by the interface under load. This can come from elastic deflection, *slop* at a joint in a mate, or misalignments in non-self-guiding docks.

For self-reconfiguring systems, attaching and detaching is automatic and the area of acceptance is required to be much larger than typical manually assembled modular systems. This is in large part because the cost of precision is high and bond precision/stiffness is typically small. The ideal mechanical interface has high bond strength, high acceptance range and high bond stiffness, all of which comes with increased weight, size, and cost. However, different styles of reconfiguration emphasize different aspects.

Lattice reconfiguration systems do not need high acceptance range as modules move in highly constrained manner when reconfiguring, usually with one degree of freedom from one lattice position to neighboring one. However, high bond stiffness is emphasized, as these modules do not often have actuated degrees of freedom to compensate for errors in position. For example, a sequence of modules in a lattice that are forming a loop might not close properly because one section of modules may sag under gravity. This sag was noticeable in the 3-D lattice module [22.82].

Chain reconfiguration modules can deal with lower bond stiffness since they usually have degrees of freedom that can compensate for errors in deflection. However, docking even with this compensation is not trivial and most systems require as large an acceptance range as possible.

Bond strength for systems that reconfigure tend to be much lower than the one would find in a manually assembled system. For example, magnetic bonding methods are easy to implement, typically have good precision and wide range of acceptance, but are weak. Manual systems that are bolted together have high bond strength, but are very complicated if they are made automatic with a very small acceptance range. The variability in bond strength versus the stiffness in joints can be used to vary the overall stiffness of a conglomerate system [22.80]. This variable stiffness can be used to make a compliant material that can conform to surfaces or a stiff material that will not bend under load.

Depending on the task, bond strength may not be required to be high. For example, in mobile reconfiguration systems that primarily move on the ground, e.g., forming trains such as *millibot* [22.67] or *swarms* [22.38], the worst case situation is when the conglomerate tries to cross a gap with modules cantilevered over the gap. Otherwise, the conglomerate is nominally supported at all times and the bond only sees dynamic friction and inertial forces. But in general, higher bond strength does not hurt and keeps a system from falling apart from static or dynamic loads.

Power and Communications Interface

Power and communications interfaces are typically electrical, though in the case of the related work in quick-change end effectors, pneumatics is often supplied too. In any case, the primary parameters of concern are the number of lines that must be transmitted between the interface, and the type of lines (e.g., pneumatic, high voltage, high current, fiber optic, etc.). The type of line will determine the size (which usually limits the number of lines) as well as the precision alignment required at the interface.

In self-reconfigurable systems, the typical interface passes electric power and a separate electronics communications bus on which all modules talk, though they are sometimes combined [22.77]. A key consideration for electronic lines is not only that the correct lines have good contact when mated, but also that wrong contacts do not touch during the docking process due to position error. For most self-reconfiguring systems maximizing the tolerance to position errors is the most important [22.56].

Communication between modules is essential for any self-reconfigurable system, whether the control is distributed or centralized. There are primarily two forms of communication: a global communications bus (any module talks to any other module) or a local communications medium (modules only talk to their neighbors through the modular interface.)

These two forms have different implications. Global buses are typically much faster than local methods of equivalent cost. There are no issues of varied latency between modules as with local methods. One problem with global buses is that there is no mechanism for modules communicating with each other to know the relative physical position of each module. Local methods get this for free. Local methods, can also emulate global architectures by message passing and routing. Local methods are also more robust to physical errors (such as shorting a communications line to power) containing failures to locally. However, they are just as

Avidbots Ex1016
Page 578 of 2154

Table 22.1 Characteristics of used communication protocols

Method	Speed (kbs)	Range (m)	Power (mW)	Address space	Rel. cost	Notes
Ethernet/CAT	1k–1m	100's	500	≈4 billion	$$$$	Complex stack (IPV4)
CANbus	10–1000	100	≈ 200	≈ 2000	$$$	Robust (base frame)
Bluetooth	1k–3k	10	≈ 100	≈1 million	$$$	3 s wake-up
BLE (wibree)	1k	10	10	≈1 million	$$	Low power Bluetooth
Zigbee	20–250	10–75	≈ 30	64 bit	$$	Wakeup 15 ms
RS485	100–10 k	1200	5	256	$	Robust elec. protoc
SPI	1–12 k	≈ 10	5	Out of band	$	Simple sync 5 wire

vulnerable to software errors such as flooding a network with garbage messages.

There are many choices for global communication protocols falling into two categories: wired and wireless. Wired is the most common and has many possible protocols, mostly requiring that the bus is multidrop. Popular wired protocols include Ethernet, EtherCAT, RS-485, SPI, and CANbus. Popular wireless protocols include Zigbee, Bluetooth, and 802.11. Important characteristics of communication protocols include the speed, real-time aspects, address space, and cost. A comparison is shown in Table 22.1 with typical values of common implementation.

The importance of the speed depends (or often dictates) the architecture of distributed control. Communications to individual modules can range from 100 times a second (for direct remote control) to several times per minute for higher level behavior control (e.g., hormone control [22.50]). Many protocols have automatic recovery from bad packets, which has obvious importance. Ethernet uses random backoff retransmission after a collision, which makes the protocol non-deterministic. This makes it difficult to have guarantees of real-time performance. The protocol is very fast so it is possible to ignore this as messages will get through with small latency with high probability. EtherCAT is Ethernet for Control Automation Technology that is better suited for real-time control. CANbus (Control Area Network bus) is a well-established bus used in the automotive technology that can be real-time, and robust, though it is typically slower than Ethernet or EtherCAT.

22.4 Conclusion and Further Reading

After more than a quarter century of research and development in modular robotics, a number of modular robotic systems have successfully entered the industry automation, education and entertainment markets. These successful modular robotic systems indicate that modular design offers advantages in the areas of product variety, application variety and creativity. However, the cost of such systems, no matter at the module level or at the system level, has room to be reduced to help mass adoption.

The cost structure of a modular robot is closely linked with the design of the individual modules and the systemic architecture to be conceived as well as the market demand and expectation. From the development history of LEGO bricks to the Schunk Powercube modules, it is clear that module design functions are becoming simpler (reducing cost) and increasing variability (increasing the user base). Hence, for reconfigurable modular manipulators, mobile systems, and self-reconfigurable modular robots, focusing on system designs to meet the expectation of the end user is the current trend.

Besides system design, standardization of mechanical and communication interfaces is critical. Like other electronic and industrial products, electronic and communication interfaces for modular robots could adopt existing industry standards depending on the form factors, connecting performance, reliability, etc. The mechanical interface design normally does not have industry standard to follow due to variety in mechanical specifications on the form factor, loading capacity, rigidity of the joints. Hence, many novel mechanical connection designs can be explored for modular robots.

Future research continues to explore increasing the number of modules. As they approach hundreds and thousands, there are interesting questions that arise as modules become more tightly coupled than current efforts [22.73]. Issues include how to deal with the increased likelihood that some modular elements are not functioning completely correctly. Biologically inspired mechanisms such as intentional cell-death and cell-replacement may become a required part of very large systems.

As the modular approach has increasingly large numbers of modules, there are many more configurations and resulting capabilities from those configurations. Future research will need to address the problem of determining appropriate or optimal configurations

for arbitrary tasks. This may lead to a better understanding of robotic tasks in general.

The most recent review article on modular mobile robots can be found in [22.13]. In [22.30], the technical challenges and future of self-reconfigurable modular robots are reviewed. Interested readers may find further information on self-reconfigurable robots in [22.31].

Video-References

VIDEO 1 SMORES
available from http://handbookofrobotics.org/view-chapter/22/videodetails/1

VIDEO 2 4x4ht4a
available from http://handbookofrobotics.org/view-chapter/22/videodetails/2

VIDEO 3 M-Blocks: Momentum-driven, magnetic modular robots self-reconfiguring
available from http://handbookofrobotics.org/view-chapter/22/videodetails/3

VIDEO 5 ATRON robot showing robust and reversible execution of self-reconfiguration sequences
available from http://handbookofrobotics.org/view-chapter/22/videodetails/5

References

22.1 K. Ulrich: The role of product architecture on the manufacturing firm, Res. Policy **24**, 419–440 (1995)

22.2 C. Paredis, H.B. Brown, P. Khosla: A rapidly deployable manipulator system, Proc. IEEE ICRA (1996) pp. 1434–1439

22.3 T. Fukuda, S. Nakagawa: Dynamically reconfigurable robot system, Proc. IEEE ICRA (1988) pp. 1581–1586

22.4 R. Cohen, M. Lipton, M. Dai, B. Benhabib: Conceptual design of a modular robot, ASME J Mech. Des. **25**, 114–117 (1992)

22.5 K.H. Wurst: The conception and construction of a modular robot system, Proc. 16th Int. Symp. Ind. Robots (ISIR) (1986) pp. 37–44

22.6 D. Tesar, M.S. Butler: A generalized modular architecture for robot structures, ASME J. Manuf. Rev. **2**(2), 91–117 (1989)

22.7 T. Matsumaru: Design and control of the modular robot system: TOMMS, Proc. IEEE ICRA (1995) pp. 2125–2131

22.8 G. Yang, I.-M. Chen, W.K. Lim, S.H. Yeo: Kinematic design of modular reconfigurable in-parallel robots, Auton. Robot. **10**(1), 83–89 (2001)

22.9 I.-M. Chen: Rapid response manufacturing through a rapidly reconfigurable robotic workcell, Robotics Comput. Integr. Manuf. **17**(3), 199–213 (2001)

22.10 Schunk GmbH, Lauffen, Germany, http://www.schunk.com/

22.11 W. Wang, W. Yu, H. Zhang: JL-2: A mobile multi-robot system with docking and manipulating capabilities, Int. J. Adv. Robotic Syst. **7**(1), 9–18 (2010)

22.12 B. Li, S. Ma, J. Liu, M. Wang, T. Liu, Y. Wang: AMOEBA-I: A shape-shifting modular robot for urban search and rescue, Adv. Robotics **23**(9), 1057–1083 (2009)

22.13 P. Moubarak, P. Ben-Tzvi: Modular and reconfigurable mobile robotics, Robot. Auton. Syst. **60**, 1648–1663 (2012)

22.14 H.C.H. Chiu, M. Rubenstein, W.M. Shen: Multifunctional superbot with rolling track configuration, Proc. IEEE/RSJ IROS (2007)

22.15 I.-M. Chen, J.W. Burdick: Enumerating non-isomorphic assembly configurations of a modular robotic system, Int. J. Robotics Res. **17**(7), 702–719 (1998)

22.16 J. Liu, Y. Wang, S. Ma, Y. Li: Enumeration of the non-isomorphic configurations for a reconfigurable modular robot with square-cubic-cell modules, Int. J. Adv. Robotic Syst. **7**(4), 58–68 (2010)

22.17 H. Liu, H. Wang, S. Li, L. He: Research on the topology description and modeling method for reconfigurable modular robots, Proc. ASME/IFToMM Int. Conf. Reconfig. Mech. Robot. (ReMAR) (2009) pp. 129–135

22.18 E. Meister, A. Gutenkunst: Self-adaptive framework for modular and self-reconfigurable robotic systems, Proc. 4th Int. Conf. Adapt. Self-Adapt. Syst. Appl. (ADAPTIVE) (2012) pp. 30–37

22.19 I.-M. Chen, G. Yang, S.H. Yeo, G. Chen: Kernel for modular robot applications – Automatic modeling techniques, Int. J. Robotics Res. **18**(2), 225–242 (1999)

22.20 I.-M. Chen, C.T. Tan, G. Yang, S.H. Yeo: A local POE model for robot kinematic calibration, Mech. Mach. Theory **36**(11), 1215–1239 (2001)

22.21 J.M. Hollerbach: A recursive lagrangian formulation of manipulator dynamics and a comparative study of dynamics formulation complexity, IEEE Trans. Syst. Man Cybern. **10**, 730–736 (1980)

22.22 G. Rodriguze, A. Jain, K. Kreutz-Delgado: A spatial operator algebra for manipulator modeling and control, Int. J. Robotics Res. **10**(4), 371–381 (1991)

22.23 R. Murray, Z. Li, S. Sastry: *A Mathematical Introduction to Robotic Manipulation* (CRC, Boca Raton 1994)

Avidbots Ex1016
Page 580 of 2154

22.24 R. Featherstone: *Robot Dynamics Algorithms* (Kluwer, Dordrecht 1987)
22.25 F.C. Park, J.E. Bobrow, S.R. Ploen: A lie group formulation of robot dynamics, Int. J. Robotics Res. **14**(6), 609–618 (1995)
22.26 I.-M. Chen, J.W. Burdick: Determining task optimal modular robot assembly configurations, Proc. IEEE ICRA (1995) pp. 132–137
22.27 J. Han, W.K. Chung, Y. Youm, S.H. Kim: Task based design of modular robot manipulator using efficient genetic algorithm, Proc. IEEE ICRA (1997) pp. 507–512
22.28 C. Paredis, P.K. Khosla: Agent based design of fault tolerant manipulators for satellite docking, Proc. IEEE ICRA (1997) pp. 3473–3480
22.29 G. Yang, I.-M. Chen: Task-based optimization of modular robot configurations: Minimized degree-of-freedom approach, Mech. Mach. Theory **35**(4), 517–540 (2000)
22.30 M. Yim, W.M. Shen, B. Salemi, D. Rus, M. Moll, H. Lipson, E. Klavins, G. Chirikjian: Modular self-reconfigurable robot systems – Challenges and opportunities for the future, IEEE Robotics Autom. Mag. **14**(1), 43–52 (2007)
22.31 K. Stoy, D. Brandt, D.J. Christensen: *Self-Reconfigurable Robots – An Introduction* (MIT, Cambridge 2010)
22.32 D. Rus, M. Vona: Self-reconfiguration planning with compressible unit modules, Proc. IEEE ICRA (1999)
22.33 S. Murata, E. Yoshida, K. Tomita, H. Kurokawa, A. Kamimura, S. Kokaji: Hardware design of modular robotic system, Proc. IEEE/RSJ IROS (2000)
22.34 K. Kotay, D. Rus: Self-reconfigurable robots for navigation and manipulation, Proc. IEEE/RSJ IROS (1997)
22.35 C. Ünsal, H. Kiliççöte, P.K. Khosla: I(CES)-cubes: A modular self-reconfigurable bipartite robotic system, Proc. SPIE, Sens. Fusion Decent. Control Robotic Syst., Bellingham (1999) pp. 258–269
22.36 A. Castano, W.M. Shen, P. Will: CONRO: Towards deployable robots with inter-robots metamorphic capabilities, Auton. Robots **8**(3), 309–324 (2000)
22.37 W.H. Lee, A.C. Sanderson: Dynamic analysis and distributed control of tetrobot modular reconfigurable robotic system, Auton. Robots **10**(1), 67–82 (2001)
22.38 E. Tuci, R. Gross, V. Trianni, F. Mondada, M. Bonani, M. Dorigo: Cooperation through self-assembly in multi-robot systems, ACM Trans. Auton. Adapt. Syst. (TAAS) **1**(2), 115–150 (2006)
22.39 J.W. Suh, S.B. Homans, M. Yim: Telecubes: Mechanical design of a module for self-reconfigurable robotics, Proc. IEEE ICRA (2002)
22.40 M.W. Jorgensen, E.H. Ostergaard, H.H. Lund: Modular Atron: Modules for a self-reconfigurable robot, Proc. IEEE/RSJ IROS (2004) pp. 2068–2073
22.41 M. Hafez, M.D. Lichter, S. Dubowsky: Optimized binary modular reconfigurable robotic devices, IEEE/ASME Trans. Mechatron. **8**(1), 18–25 (2003)
22.42 P. White, V. Zykov, J. Bongard, H. Lipson: Three dimensional stochastic reconfiguration of modular robots, Proc. Robotics Sci. Syst. **1**, 161–168 (2005)
22.43 B. Kirby, J. Campbell, B. Aksak, P. Pillai, J. Hoburg, T. Mowry, S.C. Goldstein: Catoms: Moving robots without moving parts, Proc. Natl. Conf. Artif. Intell. **20**, 1730–1731 (2005)
22.44 S.C. Goldstein, J.D. Campbell, T.C. Mowry: Programmable matter, Computer **38**(6), 99–101 (2005)
22.45 A.A. Goldenberg, N.M. Kircanski, P. Kuzan, J.A. Wiercienski, R. Hui, C. Zhou: Modular, expandable and reconfigurable robot, US Patent 552 3662 (1996)
22.46 H. Lipson, J.B. Pollack: Towards continuously reconfigurable self-designing robotics, Proc. IEEE ICRA (2000)
22.47 J.E. Walter, E.M. Tsai, N. Amato: Choosing good paths for fast distributed reconfiguration of hexagonal metamorphic robots, Proc. IEEE ICRA (2002)
22.48 A. Castano, P. Will: Representing and discovering the configuration of CONRO robots, Proc. IEEE ICRA (2001) pp. 3503–3509
22.49 K. Støy, W.M. Shen, P. Will: Global locomotion from local interaction in self-reconfigurable robots, Proc. 7th Int. Conf. Intell. Auton. Syst. (IAS-7), Marina del Rey (2002)
22.50 P. Will, B. Salemi, W.M. Shen: Hormone-controlled metamorphic robots, Proc. IEEE ICRA (2001) pp. 4194–4199
22.51 A. Kamimura, S. Murata, E. Yoshida, H. Kurokawa, K. Tomita, S. Kokaji: Self-reconfigurable modular robot – Experiments on reconfiguration and locomotion, Proc. IEEE/RSJ IROS (2001) pp. 606–612
22.52 R. Fitch, D. Rus, M. Vona: Basis for self-repair robots using self-reconfiguring crystal modules, Proc. IEEE Int. Auton. Syst. (2000) pp. 903–910
22.53 G.S. Chirikjian, A. Pamecha, I. Ebert-Uphoff: Evaluating efficiency of self-reconfiguration in a class of modular robots, J. Robotic Syst. **13**(5), 317–338 (1996)
22.54 F.H. Bennett, E.G. Rieffel: Design of decentralized controllers for self-reconfigurable modular robots using genetic programming, Proc. 2nd NASA/DoD Workshop Evol. Hardw. (2000)
22.55 Z. Butler, R. Fitch, D. Rus: Distributed control for unit-compressible robots: Goal-recognition, locomotion, and splitting IEEE/ASME Trans. Mechatron. **7**(4), 418–430 (2002)
22.56 M. Yim, Y. Zhang, K. Roufas, D. Duff, C. Eldershaw: Connecting and disconnecting for chain self-reconfiguration with PolyBot, IEEE/ASME Trans. Mechatron. **7**(4), 442–451 (2002)
22.57 Y. Zhang, M.P.J. Fromherz, L.S. Crawford, Y. Shang: A general constraint-based control framework with examples in modular self-reconfigurable robots, Proc. IEEE/RSJ IROS (2002) pp. 2163–2168
22.58 H. Bojinov, A. Casal, T. Hogg: Emergent structures in modular self-reconfigurable robots, robotics and automation, Proc. IEEE ICRA (2000)
22.59 P. Pirjanian, C. Leger, E. Mumm, B. Kennedy, M. Garrett, H. Aghazarian, S. Farritor, P. Schenker: Distributed control for a modular, reconfigurable cliff robot, Proc. IEEE ICRA (2002)
22.60 K.C. Prevas, C. Unsal, M.O. Efe, P.K. Khosla: A hierarchical motion planning strategy for a uniform

Avidbots Ex1016
Page 581 of 2154

self-reconfigurable modular robotic system, Proc. IEEE ICRA (2002)
22.61 W.W. Melek, A.A. Goldenberg: Neurofuzzy control of modular and reconfigurable robots, IEEE/ASME Trans. Mechatron. **8**(3), 381–389 (2003)
22.62 M. Yim, D.G. Duff, K.D. Roufas: Polybot: A modular reconfigurable robot, Proc. IEEE ICRA (2000) pp. 514–520
22.63 M. Yim, Y. Zhang, D. Duff: Modular Reconfigurable Robots, Machines that shift their shape to suit the task at hand, IEEE Spectrum **39**(2), 30–34 (2002)
22.64 M. Yim, C. Eldershaw, Y. Zhang, D. Duff: Limbless conforming gaits with modular robots. In: *Springer Tract. Adv. Robot.*, Vol. 21, 2006) pp. 459–468
22.65 M. Yim, J. Reich, A.A. Berlin: Two approaches to distributed manipulation. In: *Distributed Manipulation*, ed. by K. Bohringer, H.H. Choset (Kluwer, Dordrecht 2000) pp. 237–261
22.66 J. Davey, N. Kwok, M. Yim: Emulating self-reconfigurable Robots – Design of the SMORES system, Proc. IEEE/RSJ IROS (2012)
22.67 H.B. Brown, J.M. Vande Weghe, C.A. Bererton, P.K. Khosla: Millibot trains for enhanced mobility, IEEE/ASME Trans. Mechatron. **7**(4), 452–461 (2002)
22.68 S. Murata, K. Kakomura, H. Kurokawa: Toward a scalable modular robotic system, IEEE Robotics Autom. Mag. **14**(4), 56–63 (2007)
22.69 M. Yim, B. Shirmohammadi, J. Sastra, M. Park, M. Dugan, C.J. Taylor: Towards robotic self-reassembly after explosion, Proc. IEEE/RSJ IROS (2007) pp. 2767–2772
22.70 M. Kutzer, M. Moses, C. Brown, M. Armand, D. Scheidt, G. Chirikjian: Design of a new independently-mobile reconfigurable modular robot, Proc. IEEE ICRA (2010)
22.71 H. Cheng, G. Ryland: Design of iMobot, an intelligent reconfigurable mobile robot with novel locomotion, Proc. IEEE ICRA (2010)
22.72 F.P. Brooks: *The Mythical Man-Month* (Addison-Wesley, Reading 1975)
22.73 M. Rubenstein, R. Nagpal: Kilobot: A robotic module for demonstrating behaviors in a large scale (2^{10} units) collective, Proc. IEEE ICRA (2010) pp. 47–51
22.74 Y. Zhang, M. Yim, C. Eldershaw, D. Duff, K. Roufas: Scalable and reconfigurable configurations and locomotion gaits for chain-type modular reconfigurable robots, Proc. IEEE Symp. Comput. Intell. Robotics Autom. (CIRA) (2003)
22.75 E. Yoshida, S. Murata, S. Kokaji, K. Tomita, H. Kurokawa: Micro self-reconfigurable modular robot using shape memory alloy, J. Robotics Mechatron. **13**(2), 212–218 (2001)
22.76 M.E. Karagozler, S.C. Goldstein, J.R. Reid: Stress-driven MEMS assembly + electrostatic forces = 1mm diameter robot, Proc. IEEE/RSJ IROS (2009)
22.77 M.E. Karagozler, J.D. Campbell, G.K. Feddore, S.C. Goldstein, H.P. Weller, B.W. Yoon: Electrostatic latching for inter-module adhesion, power transfer, and communication in modular robots, Proc. IEEE/RSJ IROS (2007)
22.78 I. O'Hara, J. Paulos, J. Davey, N. Eckenstein, N. Doshi, T. Tosun, J. Greco, Jungwon. Seo, M. Turpin, V. Kumar, M. Yim: Self-assembly of a swarm of autonomous boats into floating structures, Proc. IEEE ICRA (2014) pp. 1234–1240
22.79 L. Vasilescu, P. Varshavshaya, K. Kotay, D. Rus: Autonomous modular optical underwater robot (amour) design, prototype and feasibility study, Proc. IEEE ICRA (2005)
22.80 P.J. White, S. Revzen, C.E. Thorne, M. Yim: A general mechanical strength model for programmable matter and modular robotic structures, Robotica **29**(1), 103–121 (2011)
22.81 N. Eckenstein, M. Yim: The X-Face: An improved planar passive mechanical connector for modular self-reconfigurable robots, Proc. IEEE/RSJ IROS (2012) pp. 3073–3078
22.82 S. Murata, H. Kurokawa, E. Yoshida, K. Tomita, S. Kokaji: A 3-D self-reconfigurable structure, Proc. IEEE ICRA (1998) pp. 432–439

Avidbots Ex1016
Page 582 of 2154

Multimedia Contents

23. Biomimetic Robots

Kyu-Jin Cho, Robert Wood

Biomimetic robot designs attempt to translate biological principles into engineered systems, replacing more classical engineering solutions in order to achieve a function observed in the natural system. This chapter will focus on mechanism design for bio-inspired robots that replicate key principles from nature with novel engineering solutions. The challenges of biomimetic design include developing a deep understanding of the relevant natural system and translating this understanding into engineering design rules. This often entails the development of novel fabrication and actuation to realize the biomimetic design.

This chapter consists of four sections. In Sect. 23.1, we will define what biomimetic design entails, and contrast biomimetic robots with bio-inspired robots. In Sect. 23.2, we will discuss the fundamental components for developing a biomimetic robot. In Sect. 23.3, we will review detailed biomimetic designs that have been developed for canonical robot locomotion behaviors including flapping-wing flight, jumping, crawling, wall climbing, and swimming. In Sect. 23.4, we will discuss the enabling technologies for these biomimetic designs including material and fabrication.

23.1 **Overview** 544
23.2 **Components of Biomimetic Robot Design** 544
23.3 **Mechanisms** 545
23.3.1 Legged Crawling 545
23.3.2 Worm-Like Crawling 547
23.3.3 Snake Robots 548
23.3.4 Flapping-Wing Flight 550
23.3.5 Wall Climbing 551
23.3.6 Swimming 553
23.3.7 Jumping 555
23.3.8 Gripping and Perching 558
23.4 **Material and Fabrication** 561
23.4.1 Shape Deposition Manufacturing 561
23.4.2 Smart Composite Microstructures 562
23.4.3 Pop-Up Book MEMS 564
23.4.4 Other Fabrication Methods 565
23.5 **Conclusion** 567
Video-References 568
References 570

Biomimetics is a broad field that covers all ranges of robotics including robot structure and mechanics, actuation, perception, and autonomy. This section will focus on robots that mimic structure and movement principles found in nature to perform desired tasks in unstructured environments.

Nature frequently inspires engineers to adopt solutions from biology for application to human challenges. Robots are built to perform certain tasks, and many tasks include moving from one place to another. The various modes of locomotion in nature have inspired robots to mimic these locomotion with a goal to overcome various obstacles in the environment, and move around with the extreme agility similar to that found in nature. Mankind has engineered various modes of transportation on ground, air, and water. On ground, wheeled vehicles are the most popular choice. In air, fixed wing airplanes and helicopters with rotating blades dominate, and in water, ships, and submarines propelled by similar rotating elements are most common. In contrast, nature

has different solutions to locomotion involving moving legs, undulating fins, and flapping wings. Many of these biological locomotion mechanisms have analogies to engineered systems. Instead of using wheels, nature uses legs of different sizes, numbers, and impedances; humans and birds are bipedal, many mammals and reptiles move on four legs, and insects have six legs, and other arthropods have eight or more legs. These legs are coordinated to move on different surfaces in a stable manner. Snakes and worms move around without legs by creating waves with their body. Instead of fixed wings with jets and rotary propellers, birds, bats, and flying insects flap their wings. Fish create undulatory motions with their bodies to swim with agility far beyond conventional boats and submarines. Each of these biological locomotion modes are based on the nature's fundamental actuator: muscle. Muscles create linear motion that is coupled to the structures that generate locomotion. Therefore, these structures are closely related to the morphology of the muscles that move the structures.

23.1 Overview

The main purpose of locomotion in manmade machines is to deliver large payloads across long distances in the shortest possible time or with the minimal energy expenditure. There are roads, airports, shipyards, and other foundations that support this transportation, enabling the structural design of cars, trains, planes, and ships to focus on transportation tasks. On the other hand, locomotion in nature is primarily for survival. Animals and insects have evolved to survive in various environments. Each species uses their mode of locomotion to hunt for food, find mates, and escape danger all in unstructured natural environments. Therefore, the requirements for biological locomotion are much more complicated than human transportation systems.

Biomimetic robots try to mimic the structural characteristics and the principles of movement to be able to move around places where conventional machines or robots are unable to perform as needed. Animals can crawl on a rugged terrain at a high speed, can climb walls without a tether, can fly in cluttered environments and hover and perch as needed.

Depending on the size of the species the optimal mode of locomotion and the underlying structure is different. For example, jumping is found frequently in small insects to escape danger, since their small size makes it hard for them to escape quickly using other forms of locomotion. Jumping by a large animal is different from jumping by small insects like a flea. Larger insects or animals tend to use their legs to run or crawl to escape the danger. Large birds flap their wings at a much lower frequency, and use gliding mode whereas bees and flies beat their wings continuously at a high frequency during flight. The structure is very different, where the wings of a bird have bones, muscle, and feathers versus the muscle-less and lightweight wings of insects. Therefore, when developing biomimetic robots, the size of the target species should be taken into account.

Since nature has evolved to survive in extreme environments, there are many examples of extreme locomotion that are typically not possible with conventional engineering designs. For example, the wall climbing robot Stickybot uses the directional adhesion principle of a gecko to climb up smooth vertical walls. However, directional adhesion alone is not enough. The design has to enable even pressure distribution on the pad to make sure the pads are well in contact with the wall. These small details can be important to performance of biomimetic robots and should be considered carefully.

Since robots are commonly built to perform tasks too tedious or dangerous for humans, it is natural to adopt designs found in nature for functionality where nature has already found a solution. In fact, many robot designs in this handbook, such as legged robots and robot hands and many examples of flying robots, underwater robots, and micro robots, are biomimetic. This chapter will focus on areas of biomimetic robot designs that have been recently expanding. Legged robots and robot hands have already formed large communities and are covered in separate chapters. There are various other forms of locomotion, e.g., crawling, jumping, swimming, and flapping, that are being researched to mimic the principle of movement and the biological structure. The structure and the components of these robots will be reviewed and discussed in this chapter.

23.2 Components of Biomimetic Robot Design

Biomimetic robot design begins with an understanding of the underlying principles at work in a natural system. For the design of a biomimetic robot, the components used in building the robot are important; the structural

design is constrained or enabled by the actuator system, the materials and the fabrication methods used to build the robot.

Understanding of the principles of movement found in nature initiates the design of biomimetic robots. This can be studied by observing the kinematics and measuring forces in moving animals, or by understanding and modeling the dynamics. Substantial research on biological locomotion analyzes the kinematics of the animal. However, it is important to understand the dynamics behind the kinematics to be able to implement the dynamic principles into the robot design, which will enable the robot to perform like an animal, instead of just looking like one. *Full* et al. proposed clock-driven, mechanically self-stabilizing, compliant sprawled-posture mechanics to explain a cockroach racing seemingly effortlessly over a rough surface [23.1, 2]. *Ellington* built a system that can measure the force generated by flapping wings to understand the thrust generated by insect wings [23.3]. These studies in biology have influenced robotics researchers, and inspired them to build hexapod and flapping-wing robots.

To implement biological principles in engineered systems, components that can create performance similar to that of nature is required. Nature's biological structures are composed of various materials, e.g., tissues, bones, cuticles, flesh, and feathers. These materials are replaced with engineered materials, e.g., metal, plastics, composites, and polymers. New engineered materials are being developed that exhibit similar properties to natural materials, which will enable life-like robots. However, the performance of the biomimetic robot can be similar to nature without actually mimicking the exact structure. Therefore, it is important to decide to what level biomimicry is required. Large robots tend to be built with conventional mechanical components such as motors, joints, and linkages made of metal. The challenge is in building meso-scale robots, where the conventional mechanical components become ineffective, due to friction and other inefficiencies. As the robot becomes smaller, it is beneficial to mimic not just the kinematics seen in nature, but also the structure at the component level. Another challenge for small scales is actuation – the actuator must be as effective as the muscle even at small scales.

A number of new manufacturing methods have become available which also enable new designs that were not previously possible by more classical machining methods and *nuts-and-bolts* assembly techniques. Examples include shape deposition manufacturing and smart composite microstructures. In many cases, these new fabrication processes facilitate the construction of novel biomimetic robots beyond what was possible by conventional means.

Levels of mimicry can vary depending on the components used in the design. The robot can simply look like an animal, but not perform similarly to the mimicked animal. On the other hand, once the principle is mimicked the robot does not have to look like the animal, but it can still perform a task in a similar manner as the animal. The different ways of mimicking allows each robot to have its own characteristic. For example, a flea uses what is called a torque reversal for jumping, which is a unique method of storing and releasing large amount of energy. In nature, this method is possible with muscles and a lightweight-legged structure. To mimic the flea's locomotion, we need an actuator that is comparable to muscles and a fabrication method that allows us to build a small-scale rigid structure.

23.3 Mechanisms

Developing multilegged robot has been achieved by using specific design which enables some function or biomimetic model inspired by multilegged insects. Especially, the ability of a cockroach enabling to run on a rough surface at a high speed has inspired to develop a series of multilegged robots. They are capable of maintaining stability during locomotion at a high speed (relative to their body length).

23.3.1 Legged Crawling

Some types of cockroaches can achieve speed up to 50 body length per second and can crawl on uneven terrain, overcoming obstacles far higher than their height [23.2]. RHex (Fig 23.1a) [23.4] is one of the first robots to implement cockroach-like characteristics. It is a hexapod crawling robot with *C*-shaped legs, so it is suitable for walking on uneven terrain or large obstacles, as shown in VIDEO 400. Mini-Whegs (Fig 23.1b) [23.5] also have unusual wheel with three spokes. Because of the wheel-spoke structure, the gait passively adapts to the terrain similar to climbing cockroaches, as shown in VIDEO 401 [23.15]. Sprawlita (Fig 23.1c) [23.6] is a hexapod crawling robot that uses pneumatic actuators on each leg and passive rotary joints so that it can achieve dynamic stability. It weighs 27 kg and its speed is over 3 body length/s or 550 mm/s. iSprawl (Fig 23.1d) [23.7] is hexapod crawling robot that uses extension of each legs during crawling. This robot is driven by electric motors and

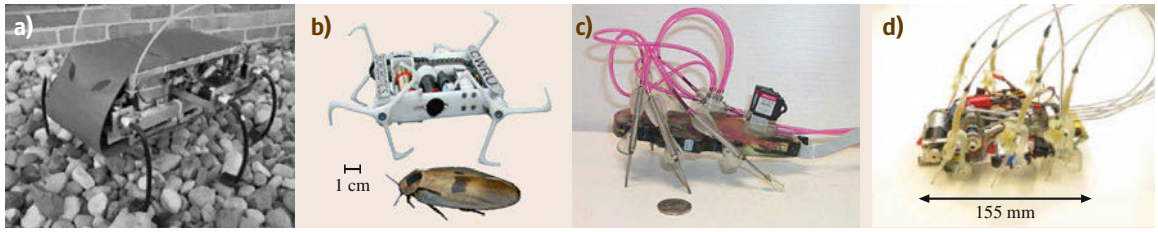


Fig.23.1a–d Crawling robots inspired by cockroach **(a)** RHex (after [23.4]), **(b)** Mini-Whegs (after [23.5]), **(c)** Sprawlita (after [23.6]), **(d)** iSprawl (after [23.7])

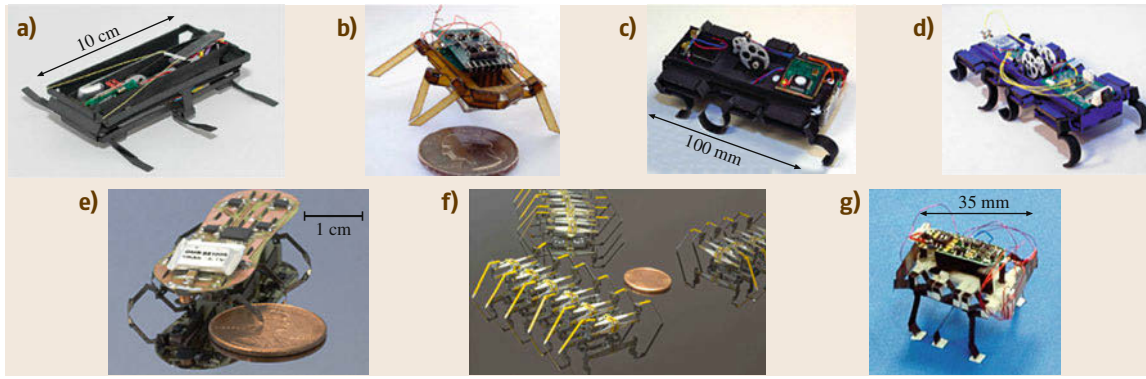


Fig.23.2a–g Crawling robots inspired by cockroach. **(a)** DASH (after [23.8]), **(b)** RoACH (after [23.9]), **(c)** DynaRoACH (after [23.10]), **(d)** OctoRoACH (after [23.11]), **(e)** HAMR3 (after [23.12]), **(f)** centipede-like modular robot (after [23.13]), **(g)** crawling robot through the integration of microrobot technologies (after [23.14])

flexible push–pull cables. Rotation of the motor causes the legs to extend or contract. It weighs 300 g and is able to crawl 15 body length/s or 2.3 m/s, as can be seen in VIDEO 403.

Cockroach-based designs would improve the performance of millimeter or centimeter scale crawling robots which face inefficiency with conventional mechanisms. DASH (Fig 23.2a) [23.8] is a hexapedal crawling robot fabricated using SCM (Smart composite manufacturing) process. This robot has only one electric motor coupled to a four-bar linkage pushing the legs in an elliptical crawling motion, as shown in VIDEO 405. DASH weighs 16.2 g and achieves speed up to 15 body length/s or 1.5 m/s. RoACH (Fig 23.2b) [23.9] is a hexapod crawling robot that imitate cockroach's alternating tripod gait. A typical cockroach gait involves the ipsilateral front leg, hind leg, and contralateral middle leg moving simultaneously. Two sets of three legs tread on a surface in turn – generating the *alternating tripod gait*. RoACH uses two shape memory alloy (SMA) wire actuators to contract the body in two orthogonal directions. Contraction of the two actuators results in motion of legs through a four-bar linkage. The legs repeatedly operate swing and stance motions according to the sequential stimulation of the two actuators and it can be seen in VIDEO 286. It weighs 2.4 g and is able to crawl 1 body length/s or 3 cm/s. DynaRoACH (Fig 23.2c) [23.10] has six legs driven by one DC motor. It has passive dynamics similar to RoACH to achieve better locomotion performance. Lift motion is achieved by a slider crank mechanism and swing motion by a four-bar mechanism. Like RHex, it uses *C*-shape legs so that the robot has lower vertical stiffness, lateral collapsibility for obstacle climbing, and more distributed ground contact. It weigh 24 g and is 100 mm long and is capable of speeds up to 14 body length/s or 1.4 m/s. OctoRoACH (Fig 23.2d) [23.11] is quite similar to DynaRoACH, but it has two motors driv-

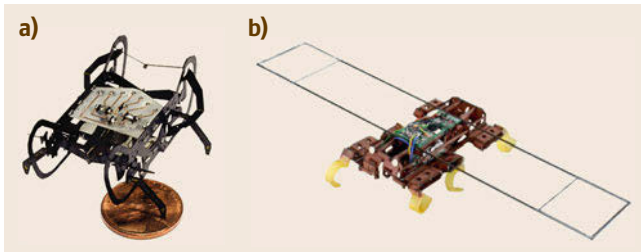


Fig.23.3a,b The latest version of crawling robots which shows improved performance **(a)** HAMRV (after [23.16]), **(b)** VelociRoACH (after [23.17])

Avidbots Ex1016
Page 586 of 2154

ing the legs on each side. It has eight legs so that it maximizes pitch stability using two motors. OctoRoACH weigh 35 g and is 100 mm long. HAMR3 (Fig 23.2e) [23.12] uses nine piezoelectric actuators. Each leg performs swing and lift motion through two decoupled piezoelectric actuator through four-bar slider-crank mechanisms and spherical five-bar mechanisms (VIDEO 406). This robot weighs 1.7 g and has speeds up to 0.9 body length/s or 4.2 cm/s (4.7 cm long robot). Especially, the HAMR is manufactured using a method inspired by pop-up books that enables fast and repeatable assembly. Multilegged robots inspired by centipede using mechanism similar to alternative tripod gaits is also developed (Fig 23.2f) [23.13]. Its several gait patterns that differ in gait frequency and phase are described in VIDEO 407. Using microrobot technologies developed until 2006, integrated structure was developed although the structure was not yet tested (Fig 23.2g) [23.14].

With more experiments and research about motion of cockroaches, the cockroaches like robots can extremely improve their performance by revision using biomimetic model. HAMRV, which is the most recent version of HAMR [23.16], can move 10.1 body length/s (44.2 cm/s) which is remarkably improved compared to preceded versions are capable of running at 0.9 body length/s. It is even capable of maneuverability and control at both low and high speeds. VelociRoACH (Fig 23.3b) [23.17] which is the latest version of RoACH can run 2.7 m/s extremely higher speed relative to previous version (VIDEO 408).

23.3.2 Worm-Like Crawling

Worm-like crawling motion can be separated into two categories: peristaltic crawling and two-anchor crawling. Worms exhibiting peristaltic crawling locomotion – such as earth worms – can move through small tunnels with limited space. Therefore, mimicry of this locomotion imposes similar characteristics to the robot and it has the potential to be used in small and harsh environment such as collapsed disaster site or inside a pipe line. which A schematic of peristaltic locomotion is shown in Fig. 23.4. By sequentially changing the volume of the body, the whole structure generates the moving motion.

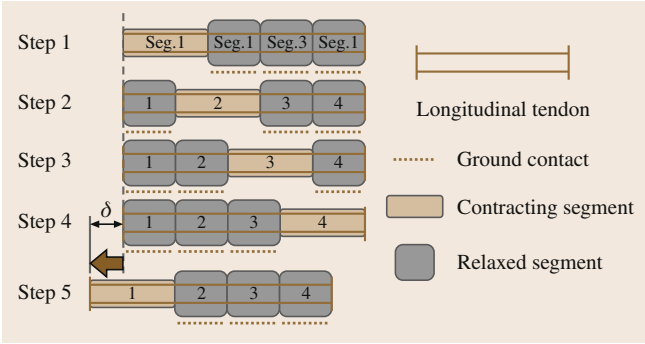


Fig. 23.4 Peristaltic locomotion (after [23.19])

The key design issue in mimicking peristaltic motion is how to create sequential volume change. Many researchers have tried various creative methods to solve this problem. *Boxerbaum* et al. built a robot with a mesh structure, and using a single motor and wire, made a partial volume change of the robot which realized a crawling motion (Fig. 23.5a) [23.18]. *Seok* et al. used a SMA coil spring actuator to change the segmented volume, and also used a mesh tube as the body structure (Fig. 23.5b) [23.19]. *Menciassi* et al. also used a SMA coil spring actuator, but they used a micro hook to enhance the friction force (Fig. 23.5c) [23.20].

Two-anchor crawling is a locomotion method used by inchworms. This locomotion mode is not fast, but it can overcome nearly any complicated topology. With an appropriate gripping method, it not only can climb vertical walls, but also can cross gaps. There are two key design issues for generating a two-anchor crawling motion: the first is how to change the shape of the waist and the second is how to anchor and unanchor to the surface. *Kotay* and *Rus* simply used an electric motor to articulate the waist motion and used an electromagnetic pad as the anchoring method (Fig. 23.6a) [23.21]. Using an electromagnetic pad, the robot can climb a steel structure. *Cheng* et al. used a tendon-driven mechanism with a compressible body and an anisotropic friction pad to generate motion (Fig. 23.6b) [23.22]. Using symmetrical or asymmetrical winding of the wire attached on the both side, the robot can make forward or steering movements. *Koh* and *Cho* used SMA coil spring actuators to control the waist motion (Fig. 23.6d) [23.23].

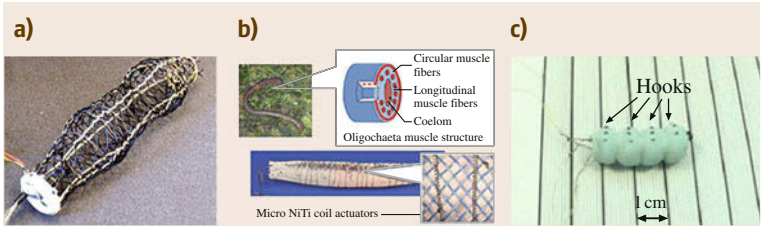


Fig. 23.5 **(a)** Robot with peristaltic motion (after [23.18]). **(b)** Meshworm robot (after [23.19]). **(c)** Biomimetic miniature robotic crawler (after [23.20])

Avidbots Ex1016
Page 587 of 2154

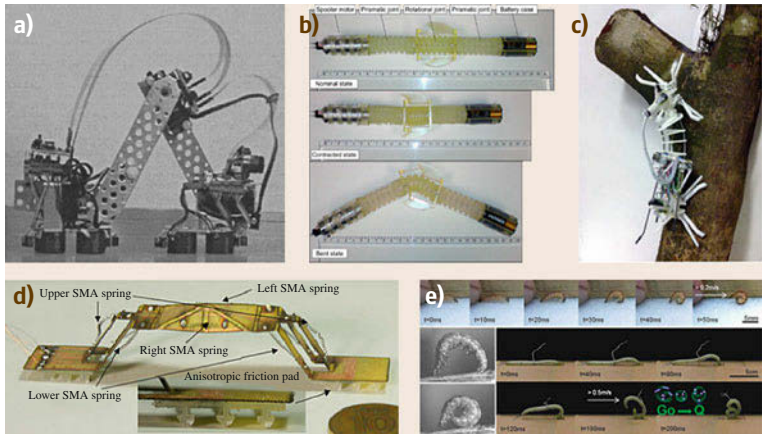


Fig.23.6a–e Two anchor crawling robot. **(a)** The inchworm robot (after [23.21]). **(b)** The soft mobile robot with thermally activated joint (after [23.22]). **(c)** Treebot (after [23.24]). **(d)** Omega shaped inchworm inspired crawling robot (after [23.23]) **(e)** GoQBot (after [23.25])

The body of the robot is made by a single sheet design with glass fiber composite, and a folding pattern designed to enable steering motion even though the robot was built by a single sheet. *Lin* et al. realized a robot with two-anchoring motion, but he also added a rolling locomotion to solve the speed limitation of previous two-anchor motion demonstrations (Fig. 23.6e) [23.25]. *Lam* and *Xu* realized another method to generate the waist motion. The robot used a backbone rod coupled to an electrical motor. By controlling the length of the backbone using the motor, the robot can control the position of the anchoring point (Fig. 23.6c) [23.24].

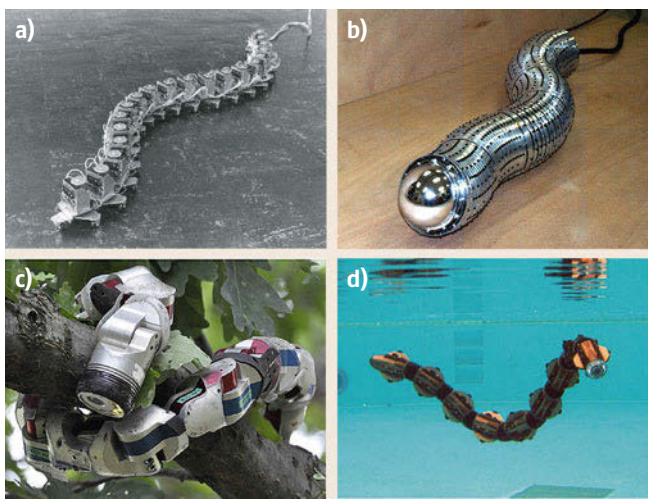


Fig.23.7a–d Three snake-like robots. **(a)** AMC-III, Shigeo Hirose, Fukushima Robotics Lab, Tokyo Institute of Technology (after [23.26]), **(b)** Slim Slime Robot II (SSR-2), Shigeo Hirose, Fukushima Robotics Lab, Tokyo Institute of Technology (after [23.27]), **(c)** Modular snake robot, Howie Choset, Biorobotics Lab, Carnegie Mellon University (after [23.28]), **(d)** AMC-R5, Shigeo Hirose, Fukushima Robotics Lab, Tokyo Institute of Technology (after [23.29])

23.3.3 Snake Robots

Studies on the locomotion of snakes began in middle of the 20th century [23.30–32]. Snakes are limbless, slender, and flexible [23.33]. Their locomotion gives them adaptability and mobility through land, uneven ground, narrow channel, pipes, and even water, and even flying between trees [23.34, 35]. An advantage of snake-like locomotion is the great versatility and freedom of movement with numerous degrees of freedom [23.36]. Additionally, snake locomotion could be efficient compared to legged animals, because there is no lifting of the center of gravity or limb acceleration [23.37]. In 1970s, *Hirose* developed a continuous locomotion model and a snake-like robot called the *Active Cord Mechanism* (ACM) [23.38]. After the Hirose's ACM robot, snake-like robots have been widely studied. In 1972, ACM-III (Fig. 23.6) was developed and it was the first robot that mimics the serpentine motion of real snakes [23.38]. The recent versions of ACM are in VIDEO 397.

Locomotion of snake-like robots can be categorized into the following different types: serpentine motion, sinus lifting, pedal wave, side-winding, spiral swimming, lateral rolling, lateral walking, mixture lean serpentine, and lift rolling motions.

The first generation of snake-like robots could only achieve motion on planar surfaces. These designs quickly evolved and current snake-like robots can go upward within narrow pipes and can climb and hold on to trees like VIDEO 393 [23.39]. To facilitate traversing large obstacles, some robots have added actuated articulation between each joint [23.40]. In addition, there are some snake-like robots that can swim in water. With a firmly waterproofed body, these robots can swim with spiral and sinusoidal locomotion patterns [23.38].

Today, mechanism design of snake-like robots can be classified with following five different types: active

Avidbots Ex1016
Page 588 of 2154

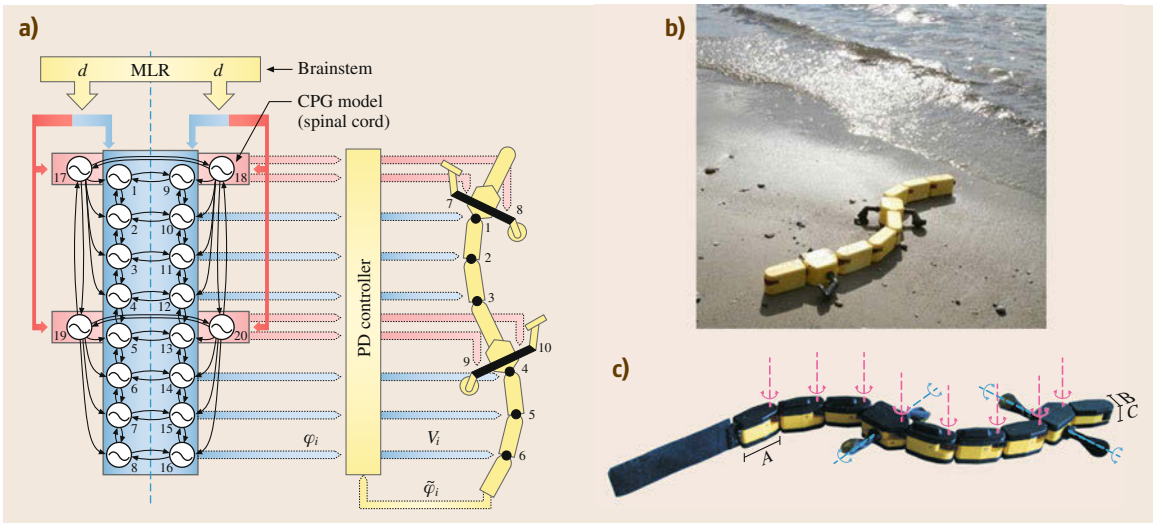


Fig. 23.8 **(a)** Configuration of the CPG model (after [23.41]). **(b)** Salamandra robotica I, Auke Jan Ijspeert, Biorobotics Lab, Ecole Polytechnique Federale de Lausanne (after [23.41]). **(c)** Salamandra robotica II, Auke Jan Ijspeert, Biorobotics Lab, Ecole Polytechnique Federale de Lausanne (after [23.42])

bending joint type, active bending and elongation joint type, active bending joint and active wheel type, passive bending joint and active wheel type, and active bending joint, and active crawler type [23.43]. Each of these types of snake-like robots consist of a number of serially connected joints. Therefore, snake-like robots are easy to modularize with their joints [23.43].

Most snake-like robots are equipped wheels that are actively or passively driven. Recently, wheelless snake-like robots have been studied [23.32]. These robots move with undulatory motion, especially lateral undulation that can be observed in real snakes. Some snake-like robots are actuated with smart materials, such as shape memory alloys and IPMCs, rather than motors [23.44]. Snake-like robots roll to avoid obstacles and interact with environment. They make waveforms with their body for propulsion. Therefore, it is important to define the configuration of the body of the robots. Measuring yaw (pitch) and roll angles are important for controlling the snake robots. Tilt sensors, accelerometers, gyroscope, and joint angle sensors are typically leveraged to control the robot [23.37]. Tactile sensors attached to the contact area or outside of the body of the snake-like robots have been used to measure surface contact forces at each joint, providing more information to the controller. Additionally, measuring contact forces could be useful for active and adaptive grasping of the snake-like robot [23.45].

However, due to the high degrees of freedom of the snake-like robots, designing controllers is not easy even for flat surface locomotion. Efforts without complex sensors and controllers are in VIDEO 392. Because of this, contrasting with the advantages of locomotion of real snakes – which can move in uneven environments – most existing snake-like robots are developed based on flat surface movement [23.46].

A major motivating application for snake-like robots is the exploration of hazardous environment that are inaccessible to humans. In particular, industrial inspection of pipes and ventilation tubes, and chemical channels are key operating environments. There is also the potential for snake-like robots as medical devices, such as minimal invasive surgery device and laparoscopy and endoscopy [23.37]. For these applications, snake-like robots require their outer skin to hermetically seal the internal components [23.40].

There are numerous challenges for snake-like robots. For greater reliability, robustness and controllability, the mechanisms and configuration of the snake-like robots could be simplified. To use the snake-like robots for exploration and inspection applications, routing of external wires for electronics and power to the robot is an important consideration. Finally, with a large number of degrees of freedom typical of snake-like robots, designing efficient control strategy is large issue [23.37].

Similar to undulatory locomotion of snakes, body of salamanders makes *S*-shape standing waves. They are capable of rapidly switching between swimming and walking locomotion. Their locomotion in aquatic and terrestrial environments is generated by a central pattern generation (CPG) and stimulation of a mesencephalic locomotor region (MLR) located in the midbrain (Fig. 23.7 and VIDEO 395) [23.41]. There were

Avidbots Ex1016
Page 589 of 2154

some efforts to produce similar swimming and walking gaits to real salamander with robotic salamanders. The salamander robots with the mathematical CPG model, DC motors, and oscillators could produce similar kinematics to real salamanders [23.42] (Fig. 23.8).

23.3.4 Flapping-Wing Flight

Flapping-wing flight is a common inspiration for biomimetic aerial robots. This is due to the agility of natural flyers such as birds, bats, and insects. Advances in the understanding of the aerodynamics of flapping-wing based on hydrodynamic theories and experimental results have provided insights into thrust production in flapping-wing animals [23.47]. From observation of the flapping motion exhibited by insect flight, the translational, and rotational motions of the wing produce lift forces at high angles of attack – beyond what is typical for fixed wing aircraft. The formation of a large vortex at the leading edge of the wing and the recapturing of shed vortices by properly timing of the swing enhance the resultant lift force [23.48]. Figure 23.9 shows the representative motion of an insect wing during flight and the vector formation of hydrodynamic forces. Such characteristics learned from nature inspires the design of wing-driving systems in flapping-wing flying robots.

Most flapping mechanisms are constructed from electromagnetic rotary motors driving crank-rocker linkages to flap the wings. Examples include the DelFly II [23.49], Robot dragonfly [23.50] and Nano Hummingbird [23.51] as shown in Fig. 23.10. As can be seen in VIDEO 402, DelFly II employs four-wing morphology where two wings on each side perform a *clapping* motion during each period. This contributes to lower power consumption and the low rocking amplitude of the fuselage. The DelFly II model used a crank mechanism such that the gear axis is perpendicular to the flying direction, for overcoming phase differences between the two sets of wings and this difference induced rotational movement on the fuselage [23.49]. In related work, the Robot Dragonfly [23.50] is inspired from a dragonfly's hovering capabilities and utilizes a tandem wing model. Each transmission link is attached to gear mechanisms that can make wing motions similar to those of a dragonfly. The Nano Hummingbird uses single pair of flapping wings as with real hummingbirds. The flapping mechanism is a dual lever, string-based flapping mechanism. Wing rotation modulation for control of the wing attack angle is achieved by the two adjustable *stops* which limit how far each wing can rotate [23.51]. Alternative mechanisms use oscillating actuators and flexure-based transmissions systems (these mechanisms are discussed in Sect. 23.4.2).

As the robot scale decreases, previous flapping mechanisms have been difficult to adapt due to manufacturing challenges and the physics of scaling. The Harvard robotic fly shows promising fabrication and design processes to build the small scale flapping air vehicle as shown in Fig. 23.11 [23.52]. Previous versions of the Harvard robotic fly had three degrees of freedom, only one of which was actuated. The angle of attach of two the wings are passively controlled by a compliant flexure joint connected to the transmission. The wing beat frequency is tuned at 110 Hz in resonance. Through advances in meso-scale manufacturing methods, the capabilities of the robotic fly have been dramatically extended and unconstrained flight

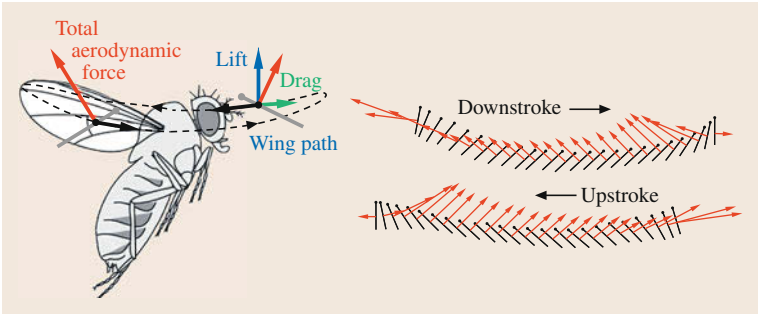


Fig. 23.9 Diagram of wing motion indicating magnitude and orientation of the aerodynamic force (after [23.48])

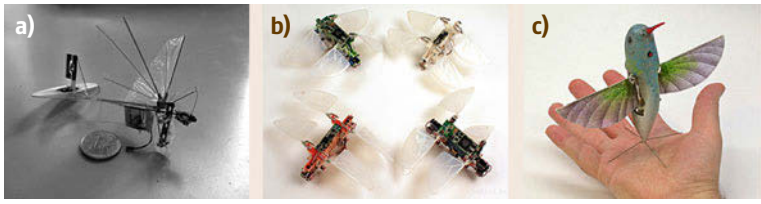


Fig. 23.10 **(a)** Delfy II (after [23.49]), **(b)** Robot Dragonfly [23.50], **(c)** Nano Hummingbird (after [23.51])

Avidbots Ex1016
Page 590 of 2154

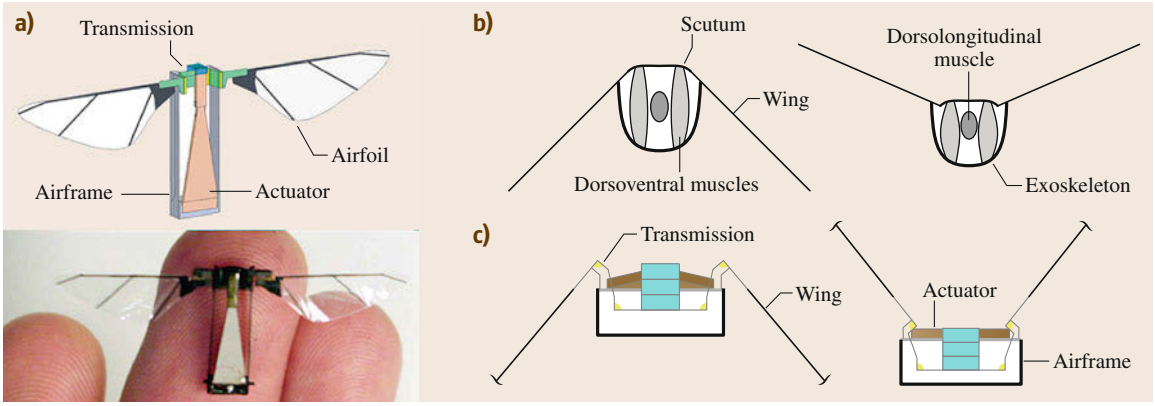


Fig. 23.11 (a) Harvard robotic fly, (b) Illustrations of thorax mechanisms, (c) and transmission (after [23.52])

has been demonstrated like VIDEO 399 in an 80 mg robot as discussed in Sect. 23.4.2.

Flapping-wing locomotion can also be extended to multimodal locomotion in combination with other mechanisms. In the DASH+Wings shown in Fig. 23.12 [23.53], the combination of wing flapping and crawling compliments each other for improving agility and stability. This hybrid robot improves performance of the maximum horizontal running speed in a factor of two and the maximum climbing incline angle by a factor of three.

23.3.5 Wall Climbing

Climbing and maneuvering on vertical surfaces present a difficult challenge. However, this locomotion mode is needed in many areas such as shipping, construction, and terrestrial locomotion in natural environment. Early attempts involved the use of suction cups, magnets, or pressure-sensitive adhesives to implement climbing. More recently, claw, spines and sticky pads inspired by nature have been used. Climbing insects and animals inspired many researchers. Insects and reptiles employ small spines that catch on fine asperities. Geckos and some spiders employ large numbers of very fine hairs that achieve adhesion based on van der Waals interaction.

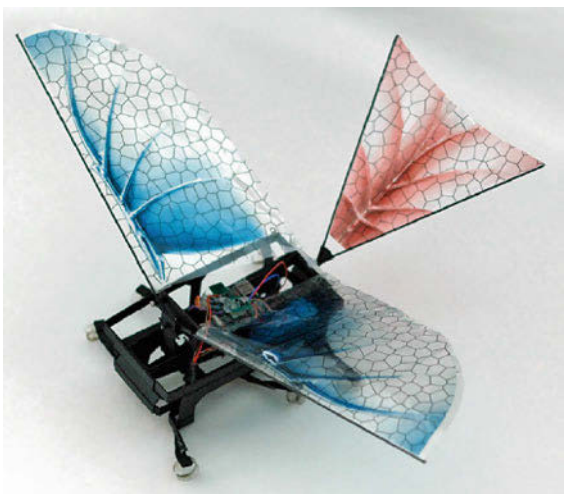


Fig. 23.12 DASH+Wings (after [23.53])

Early in the 1990s, nonbiomimetic wall climbing robots, i. e., the Ninja-1, RAMR, and Alicia were developed using suction cup. Ninja-1 attaches to a wall making use of a suction mechanism (Fig. 23.13a). The main mechanism consists of a three-dimensional (3-D) parallel link, a conduit-wire-driven parallelogram, and valve-regulated multiple sucker that enabled the robot to attach the surface with grooves [23.54]. RAMR used underactuation to remove the redundant actuators to drive the small two-legged robot [23.59]. The Alicia robot was developed for a variety of applications such as maintenance, building inspection, and safety in process and construction industries. An aspirator is used to depressurize a suction cup, so the whole robot can adhere to the wall like a standard suction cup. The Alicia3 robot use three of the Alicia II modules, allowing the whole system to better deal with obstacles on the target surface [23.60]. REST is an exceptional case that applies electromagnets instead of suction cup. It climbed only ferromagnetic wall using electromagnetic four legs with 12-DOF [23.61].

The effective wall climbing mechanisms of animals and insects have inspired development of biomimetic wall climbing robots. Typical robots utilizing bio-inspired spines found in climbing insects and cockroach are Spinybot and RiSE. Spinybot in VIDEO 388 climbs hard vertical surfaces including concrete, brick, stucco, and masonry with compliant microspine arrays (Fig. 23.13b). It can exploit small asperities (bumps or pits) on the surface. The sequence of motions is

Avidbots Ex1016
Page 591 of 2154

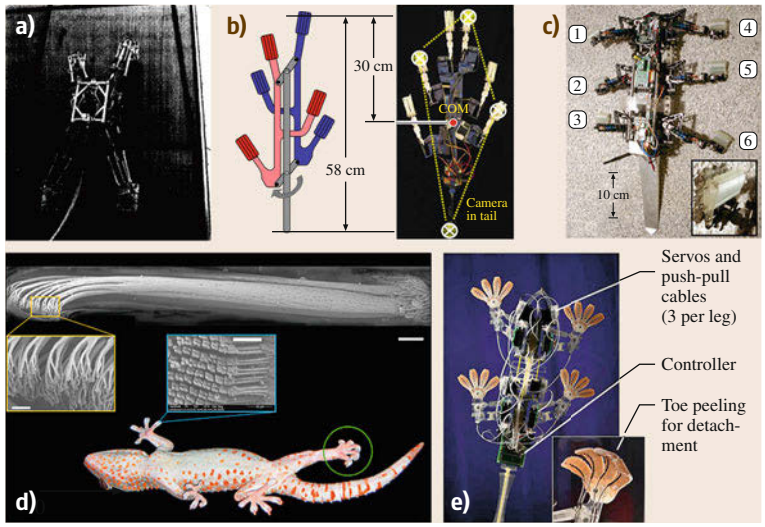


Fig. 23.13 **(a)** Ninja-1 (after [23.54]), **(b)** Spinybot II (after [23.55]), **(c)** RiSE (after [23.56]), **(d)** Gecko adhesive system (after [23.57]), **(e)** Stickybot (after [23.58])

accomplished using an under-actuated mechanism consisting of a single rotary RC servo motor and several spines independently engaging asperities on the concrete surface [23.55]. RiSE is a hexapod robot capable of locomotion on ground and vertical surfaces such as brick, stucco, crushed stone, and wood, as shown in VIDEO 390 (Fig. 23.13c) [23.56]. To climb a vertical wall, it uses microspines inspired by cockroach's tarsus structure. In addition, it held its center of mass close to surface to minimize the pitch-back moment. RiSE also employed a static tail to reduce disparity from the pull-in forces experienced by the different legs [23.56, 62].

Insects and geckos can provide inspiration for novel adhesive technology and for the locomotion mechanisms employed during climbing. Geckos are able to climb rapidly up smooth vertical surfaces and biologists reveal that a gecko's foot has nearly five hundred thousand keratinous hairs or setae. And measured adhesive force values show that individual seta operate by van der Waals forces. The gecko's toe uncurling and peeling suggests that two aspects of setal function increase effectiveness [23.63]. The subsequent study shows that the linear relation between adhesion and shear force is consistent with a critical angle of release in live gecko toes and isolated setal arrays (Fig. 23.13d). And the frictional adhesion model provides an explanation for the very low detachment forces observed in climbing geckos that does not depend on toe peeling [23.57].

Stickybot, Mini-Whegs, Geckobot, and Waalbot are prototypical robots that leverage biomimetic dry adhesives. Stickybot climbs smooth vertical surfaces such as glass, plastic, and ceramic tile at 4 cm/s, as shown in VIDEO 389 (Fig. 23.13e). The robot employs several design principles adapted from the gecko including a hierarchy of compliant structures, directional adhesion. The undersides of Stickybot's toes are covered with arrays of small, angled polymer stalks. They readily adhere when pulled tangentially from the tips of the toes toward the ankles. When pulled in the opposite direction, they release [23.58]. Mini-Whegs uses wheel-legs with compliant, adhesive feet for climbing. The foot motion mimics the foot kinematics of insects, in order to test new bio-inspired adhesive technologies and novel, reusable insect-inspired polymer (polyvinylsiloxane) [23.67]. Geckobot has kinematics similar to a gecko's climbing gait. It uses a novel peeling mechanism of the elastomer adhesive pads, steering mechanisms and an active tail for robust and agile climbing [23.68, 69]. Waalbot used two actuated legs with rotary motion and two passive revolute joints at each foot. The robot has ability to climb on nonsmooth surfaces as well as on inverted smooth surfaces using gecko-like fibrillar adhesives and passive peeling. It is also capable of plane-to-plane transitions and steering to avoid obstacles [23.70].

Different approaches for climbing include electroadhesion. Electroadhesives use a novel clamping technology called compliant electroadhesion – a form of electrically controllable adhesion. This involves inducing electrostatic charges on a wall substrate using a power supply connected to compliant pads situated on the moving robot. This generates high clamping forces that are around 0.2–1.4 N supported for a one square centimeter clamp area, depending on the substrate. Regarding power considerations for electroadhesion, assuming 50% conversion efficiency, in the worst-case scenario, two AAA batteries weighing 7.6 g each can hold up a robot in *perch* mode for almost one year. Electroadhesion combined with a conventional wheeled robot results in inchworm-style wall climbing [23.79].

Avidbots Ex1016
Page 592 of 2154

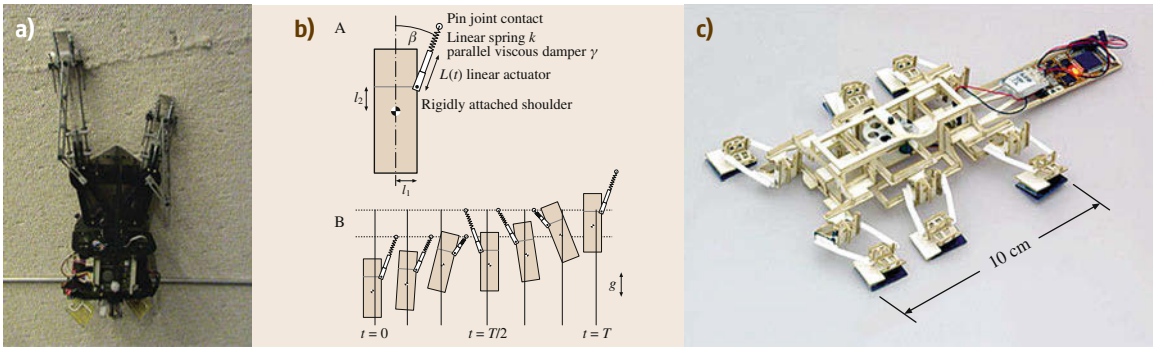


Fig. 23.14 **(a)** DynoClimber (after [23.64]). **(b)** A dynamic template for climbing (after [23.65]). **(c)** CLASH with gecko pad (after [23.66])

Dynamic wall climbing is the next challenge for wall climbing robots because previous robots were slow and in most instances restricted to targeted surfaces. For dynamical climbing originating in biology, pendulous climbing model was proposed (Fig. 23.14b). This model abstracts remarkable similarities in dynamic wall scaling behaviour exhibited by radically different animal species such as cockroaches and geckos [23.65]. The findings suggest that animals employ large lateral in-pulling forces and body rotations to achieve fast, self-stabilizing gaits. DynoClimber displays the feasibility of adapting the dynamics to robot that runs vertically upward (Fig. 23.14a). A novel bi-pedal dynamic climber can scale a vertical wall fast accompanying while achieving dynamic stability. For dynamic climbing, this robot consists of a DC motor, a crank slider mechanism and passive-wrist springs so its climbing at speeds 0.67 m/s (1.5 body lengths/s) [23.64]. The climbing robot CLASH has modified DASH platform but it actuates in horizontal direction to reduce height (7 mm from robot bottom). One of the key points is passive foot mechanism. When climbing upward, the foot hangs its spines on the surface and then retracts passively. This increases the shear and normal forces and enables climbing on loose cloth at 15 cm/s speed, as shown in VIDEO 391 [23.80]. The next version of CLASH has a foot that consists of an 18×15 mm pad of microfabricated PDMS (polydimethylsiloxane) ridges inspired by gecko feet (Fig. 23.14c). The ankle is an isosceles-trapezoid four-bar that creates a remote center-of-motion. This mechanism allows the foot to make coplanar contact with the surface and reduces roll peeling moments [23.66].

23.3.6 Swimming

Underwater vehicles have been made to achieve marine explorations, surveillance, and environmental monitoring. Most underwater vehicles employ propellers for propulsion and these vehicles have shown great performance with respect to the cost of transport. However, efficiency and manoeuvrability in confined areas is problematic for most surface or underwater vehicles. Moreover, propeller-driven vehicles risk tangling when moving thourgh environments with debris and vegetation. To resolve these issues, researchers have tried to replace the conventional rotary propellers with undulatory movement inspired by fish (Fig. 23.15).

The undulatory movement of fish provides two main advantages – manoeuvrability in confined areas and high propulsive efficiency. The main difference between existing propeller and undulatory movement is turning radius and speed. Fish can turn with a radius 1/10 of their body length, while propeller-driven ships require a much larger radius. Accordingly, the turning speed of fish is much faster than ships. Beyond manoeuvrability, the driving efficiency in biological swimmers also show more improvement over man-made systems [23.81].

To achieve fish-like swimming motion, various mechanisms have been employed such as linkage mechanisms and compliant mechanisms. *Barrett* first proposed a RoboTuna by using six servo motors and eight linkages [23.71]. *Morikawa* et al. built a robotic fish mimicking the caudal musculo-skeletal structure of a tuna with two rubber pneumatic artificial muscles and a multijoint bending mechanism [23.82]. A robotic dolphin was designed with four links and six servo motors to mimic the dorsoventral movement of a real dolphin [23.83]. Low developed a fish robot to generate arbitrary undulating waveforms, by connecting ten servo motors in series by linking them with sliders [23.74]. *Liu* and *Hu* developed a robotic fish mimicking the body motion of carangiform fish by using three servo motors on each joint, as shown in VIDEO 431 [23.72]. Yang et al. presented Ichthus V5.5 by using 3-DOF serial link-mechanism with servo motors on each joint for propulsion, as shown in VIDEO 432. Ichthus V5.5 has

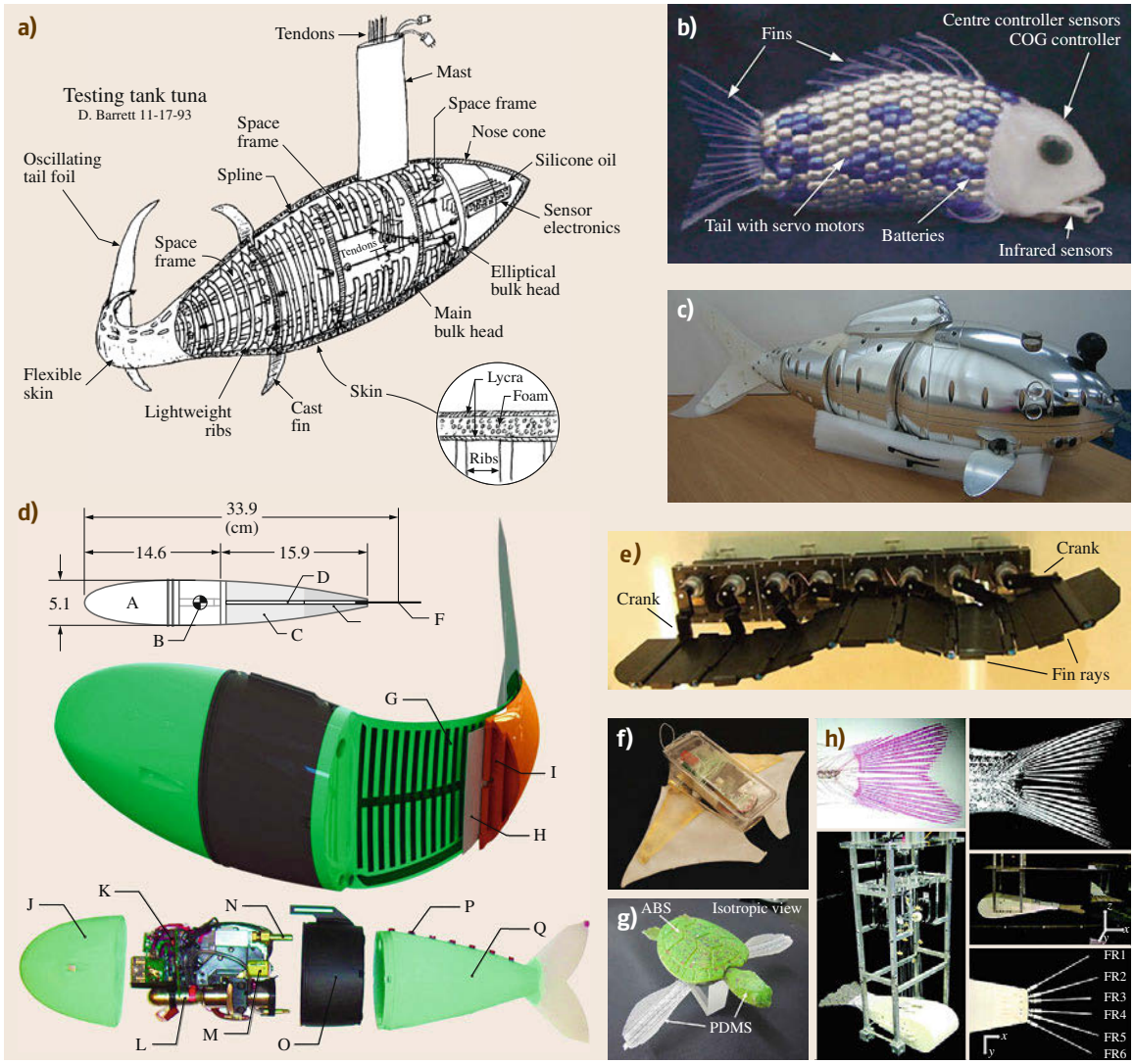


Fig.23.15a–h Design of robotic fin **(a)** Robo Tuna (after [23.71]), **(b)** G9 serise robotic fish (after [23.72]), **(c)** Ichthus (after [23.73]), **(d)** Undulating robotic fin rays (after [23.74]), **(e)** Soft robotic fish (after [23.75]), **(f)** Robotic mata ray (after [23.76]), **(g)** Turtle-like swimming robot (after [23.77]), **(h)** Robotic fin rays (after [23.78])

several sensors to navigate autonomously in the real environment such as rivers [23.73].

Beyond linkage mechanisms, several researchers have employed compliant materials in their designs to make the undulatory motion without complicated linkage structures. *Salumäe* and *Kruusmaa* implemented swimming kinematics of a trout by simply adjusting the compliance of a flexible fin with a single actuator [23.84]. *Alvarado* and *Youcef-Toumi* designed a robotic fish with a simple and robust mechanism, using a compliant body that was approximated by a continuous cantilever beam to generate a fish-like oscillating motions [23.85]. This simple design achieved biomimetic locomotion using only one servomotor, while most other robotic fish use several motors to achieve biomimetic modes of swimming. *Marchese* et al. employed a compliant body with embedded actuators and used a novel fluidic actuation system that drives body motion, as shown in VIDEO 433 [23.75]. Park et al. presented a guideline for optimizing the fin to maximize the thrust generated by a compliant fin. The *half-pi phase delay* condition describes the condition that the thrust is maximized regardless of the shape of the fin, driving frequency, and amplitude. They also presented a variable-stiffness flapping mechanism to improve the performance of a compliant fin while

Avidbots Ex1016
Page 594 of 2154



Fig.23.16a,b Escapement cam mechanism **(a)** Grillo (Ver.1) (after [23.86]) and **(b)** 7 g jumping robot (after [23.87])

the operating conditions change [23.88]. Tendons were used to vary the stiffness and the attachment point is determined based on the anatomy of a dolphin's fluke. Several fish robots that use smart actuators to create undulating motion have also been investigated [23.89]. *Wang* et al. embedded shape memory alloy (SMA) wire actuator to create flexible bending and investigated the musculature of a cuttlefish fin to aid the design of the biomimetic fin [23.90]. *Chen* et al. mimicked manta ray by using ionic polymer-metal composite (IPMC) as artificial muscles. They embedded an IPMC muscle in each pectoral fin and a passive PDMS membrane to lead to an undulatory flapping motion on the fin, as shown in VIDEO 434 [23.76]. *Kim* et al. [23.77] used a smart soft composite (SSC) structure to generate bending and twisting motions in a simple, lightweight structure. *Lauder* et al. designed a robotic fish caudal fin with six independently moving fin rays based on the anatomy of bluegill sunfish and presented that the cupping motion produced greater thrust than others such as *W*-shaped, undulation, and rolling [23.78]. They used five different sets of fin rays and measured thrust by varying the motion program. In addition, *Lauder* et al. used a flexible plastic foil to explore the effects of changing swimming speed, foil length, and shape of the foil-trailing edge on undulatory locomotion [23.91].

23.3.7 Jumping

In nature, many animals use jumping as a locomotion strategy. Jumping has the advantages of overcoming large obstacles and avoiding predators quickly and increasing the chances of survival. Robots also experience challenges in overcoming obstacles larger than the characteristic dimension of the robot. To find solutions for this, many researchers have developed jumping robots inspired by nature.

The jumping process requires large amounts of energy to be released instantaneously. However, muscle has limited reaction speed – achieving a maximum acceleration of $15\,\mathrm{m\,s^{-2}}$. Therefore, many small creatures, such as insects, have adapted special elastomers for energy storage to generate large accelerations instead of using muscles. On the other hand, most large creatures, such as human that have relatively long legs, primarily use large muscles that can generate sufficient large force to swing long legs quickly.

In small-scale jumping, to achieve large instantaneous acceleration, the jumping process has two steps: 1) slow energy storage and 2) rapid release of the stored energy. Escapement cam mechanism is widely used to achieve these two steps. It consists of a spring, a radius varying cam, a motor, and a gearbox for torque amplification. The motor rotates the cam slowly, but powerfully, toward the direction of compressing or extending the spring. At the final portion of the cycle, the cam's radius returns to the initial state instantly, releasing the spring causing the robot to jump. Grillo (Ver.1) (in VIDEO 278) [23.86] and a similar 7 g jumping robot (in VIDEO 279) [23.87] use this escapement cam mechanism (Fig. 23.16).

A toothless gear mechanism is similar to the escapement cam mechanism. The main differences are that the toothless gear mechanism uses an incomplete gear instead of a change in the cam shape. The motor actuates the incomplete gear, and the gear actuates a transmission to compress or extend a spring. When the

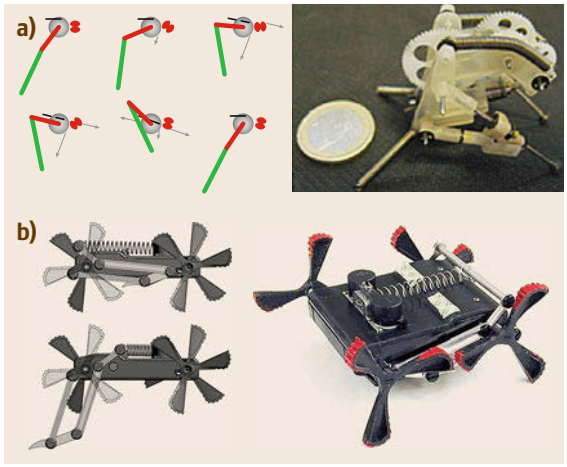


Fig.23.17a,b Toothless gear mechanism **(a)** Grillo (Ver.2) (after [23.92]), **(b)** Mini-Wheg (after [23.93])

Avidbots Ex1016
Page 595 of 2154

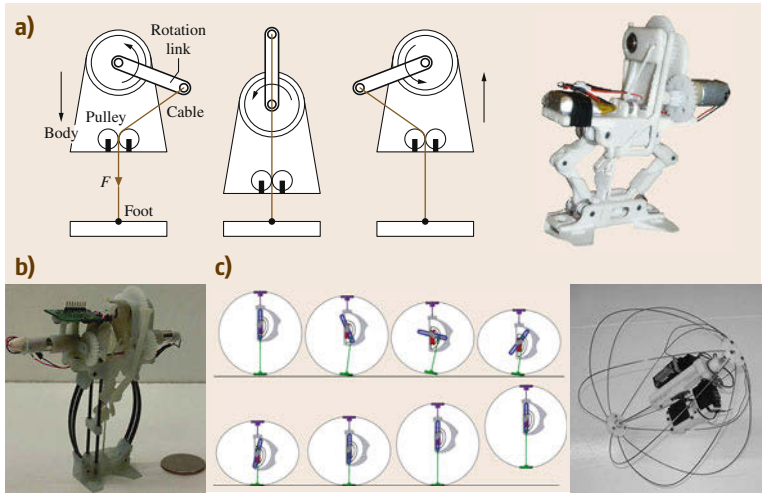


Fig.23.18a–c Other escapement mechanism **(a)** MSU Jumper (after [23.94]), **(b)** MSU Jump-runner (after [23.95]), **(c)** Jollbot (after [23.96])

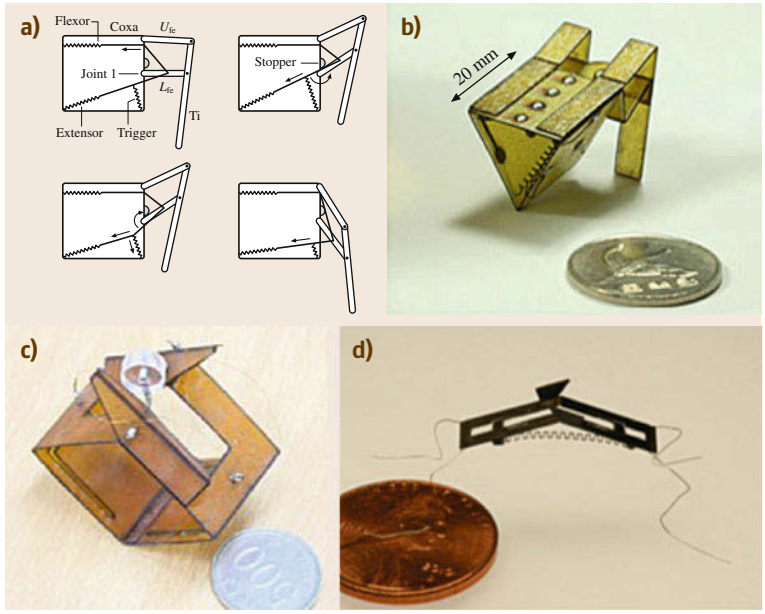


Fig. 23.19 **(a)** Torque-reversal mechanism, **(b)** Flea-inspired catapult mechanism (after [23.97]), **(c)** Simplified flea-inspired catapult mechanism (after [23.98]), **(d)** Jumping robotic insect (after [23.99])

transmission passes the toothless gear part, the transmission returns to the initial position and the stored energy is released instantly. Examples of robots using toothless gear mechanisms include Mini-Whegs [23.93] and Grillo (Ver.2) [23.92] (Fig. 23.17).

These two click mechanisms, the escapement cam mechanism and the toothless gear mechanism, are commonly used for jumping mechanisms, but other methods have also been developed (Fig. 23.18 and VIDEO 280). The catapult mechanism of MSU Jumper [23.94] and MSU Jump-Runner [23.95] is similar to the escapement cam mechanism, except for the absence of a cam. Instead of using a cam, it uses a one-way bearing. This mechanism can be separated in two parts based on the critical point in the jump cycle. Before passing the critical point, the one-way bearing cannot rotate freely, so it rotates to the direction of energy storage. On the other hand, after passing that point, it can rotate freely and release the stored energy for jumping. The catapult mechanism in Jollbot [23.96] is similar to the mechanism in the MSU Jumper. Its structure's slit acts like a one-way bearing.

A flea-inspired catapult mechanism [23.97] is different from aforementioned catapult mechanisms (Fig. 23.19 and VIDEO 281). It consists of three SMA coil springs: (1) Flexor, (2) Extensor, and (3) Trigger. The SMA is activated by heat induced from applied current. The flea catapult mechanism begins by activating

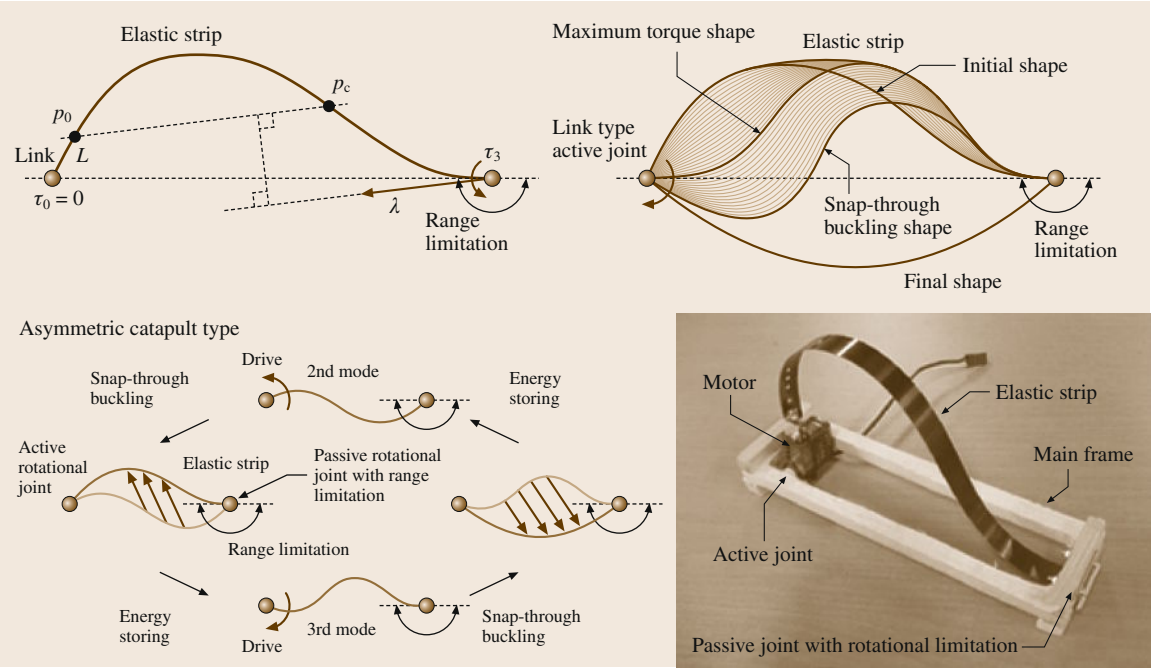


Fig. 23.20 An asymmetric robotic catapult jumping robot (after [23.100])

the flexor that folds the leg. Then, the extensor is activated. Because the direction of the torque generated by the extensor force is in the folding direction, the leg does not move and energy is stored in the extensor SMA coil spring. After energy is storage, activation of the trigger attached to the extensor pulls the extensor until it passes the joint. As a result, the direction of the torque generated by the extensor is reversed and the robot starts to jump. This mechanism uses muscle-like actuators to create the torque reversal mechanism, which enables simple design. Variation of the torque reversal mechanisms have been developed with lesser number of actuators but maintaining the same biological principle: the simplified flea-inspired catapult mechanism [23.98] and the jumping robotic insect [23.99].

An asymmetric robotic catapult jumping robot [23.100] also has a unique catapult mechanism (Fig. 23.20). It utilizes buckling in a compliant beam to jump. It consists of a main frame, an elastic strip and a motor. The elastic strip is connected to the main frame with a free rotational joint that can rotate from 0° to 180° and is connected to a motor that can control the rotation angle. When one of the elastic strip's ends is fixed to the main frame, adjusting the angle of the other end can be in a snap-through buckling shape. This can produce bidirectional jumps, but the buckling modes are different on either side.

Jumping mechanisms for microrobots [23.101] have been built using microelectromechanical systems (MEMS) manufacturing methods to create a silicon body, silicon leg, and a series of PDMS springs (Fig. 23.21a). This mechanism consists of two rigid bodies connected by PDMS springs and is activated by an external force. The robot includes only the mechanisms required to demonstrate a jump. The actuation combined elastomer mechanism is shown in Fig. 23.21b [23.101]. It used chevron actuators. These actuators are used to linearly pull and release the PDMS spring embedded into an etched silicon structure for jumping. The mechanism's PDMS springs are designed

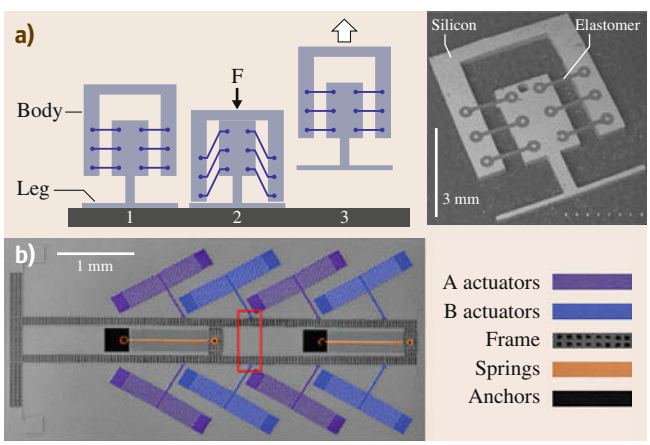


Fig. 23.21 **(a)** Microrobot (after [23.101]), **(b)** a colored SEM image of the actuated mechanism (after [23.101])

Avidbots Ex1016
Page 597 of 2154

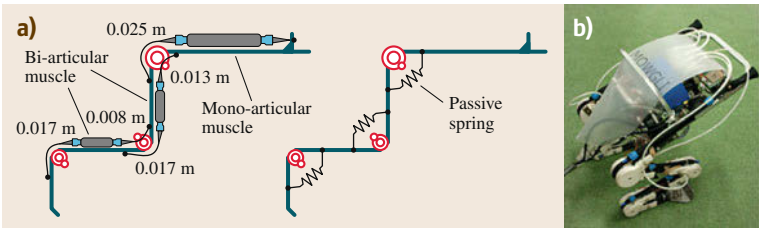


Fig. 23.22 (a) Pneumatic artificial muscles arranged like bi-articular muscle, (b) Mowgli (after [23.102])

to be similar to resilin – the elastomer that appears in insects [23.101].

In large-scale jumping, to overcome muscle's limited speed, large animals use their long legs or special arrangements of muscles and bones such as *bi-articular muscles*. They are muscles that work on two joints. In a mechanical linkage that is composed of bi-articular muscle and bones, the two joints affect each other. During the jumping process, these conditions can be helpful for generating the optimum force. The large jumping robot *Mowgli* uses long legs and pneumatic artificial muscles that are arranged like bi-articular muscle and it can be seen in Fig. 23.22 and in VIDEO 285 [23.102].

Specifications of the jumping robots are summarized in Table 23.1.

23.3.8 Gripping and Perching

In nature, insects and animals climb various kinds of terrains – from flat and smooth to wavy and rugged surfaces. Some animals evolved in a way that passively adapts to unstructured environments to reduce the energy consumption and control complexity. From the view point of robotics, these properties have the potential to increase energy efficiency and reduce system complexity. Therefore, many researchers have employed such mechanism to gripping and perching devices.

Hawkes et al. developed a mechanism that allows large patches of directional dry adhesives to conform to the topology of the surfaces they are in contact with [23.103]. The mechanism uses a rigid tile supported by a compliant material loaded by an inextensible tendon – inspired by the tendon system and the fluid-filled sinus in gecko toes. This mechanism permits the adhesive to make full contact with the surface and have uniform loading despite significant errors in alignment. *Hawkes* et al. also developed a gasper for landing of microair vehicles and grappling objects in space using gecko-inspired directional adhesives, as shown in VIDEO 413 [23.104] (Fig. 23.23).

There are several devices and robots that employ microspines for grasping rough surfaces easily seen in nature. *Kim* et al. proposed arrays of miniature spines that catch opportunistically on surface asperities [23.55]. *Desbiens* et al. proposed a small and unmanned aircraft that can land, perch and take off from vertical surfaces, as shown in VIDEO 412 [23.105] inspired by squirrels that reduce their horizontal velocity up to 60% prior to impact to distribute impact over all four limbs. *Spenko* et al. developed a hexapedal climbing robot using rows of toes having microspine [23.56]. *Parness* et al. also employed 16 carriages, each of which contains 16 microspines that conform to mm-scale and below, as shown in VIDEO 414 [23.106] (Fig. 23.24).

Trimmer et al. employed a passive gripping method found in caterpillars [23.107]. Caterpillars use their re-

Table 23.1 Specifications of the jumping robots

Robot	Actuator	Length	Weight	Jumping height	Initial velocity
7 g jumping robot [23.87]	Motor	5 cm	7 g	1.4 m	5.9 m/s
Grillo(Ver.1) [23.86]	Motor	5 cm	15 g	–	1.5 m/s
Grillo(Ver.2) [23.92]	Motor	3 cm	10 g	–	3.6 m/s
Mini-Wheg [23.93]	Motor	9–10 cm	90–190 g	0.18 m	–
MSU Jumper [23.94]	Motor	6.5 cm	23.5 g	0.87 m	–
MSU Jump-runner [23.95]	Motor	9 cm	25 g	1.43 m	–
Jollbot [23.96]	Motor	30 cm	465 g	0.218 m	–
Flea-inspired catapult mechanism [23.97]	SMA	2 cm	1.1 g	0.64 m	4.4 m/s
Simplified flea-inspired jumping mechanism [23.98]	SMA	3 cm	2.3 g	1.2 m	7 m/s
Jumping robotic insect [23.99]	SMA	2 cm	0.034 g	0.3 m	2.7 m/s
An asymmetric robotic catapult jumping robot [23.100]	Motor	17 cm	30 g	0.2 m	–
Microrobot [23.101]	None	0.4 cm	0.008 g	0.32 m	3 m/s
Mowgli [23.102]	Pneumatic	1 m	3 kg	0.4 m	–

Avidbots Ex1016
Page 598 of 2154

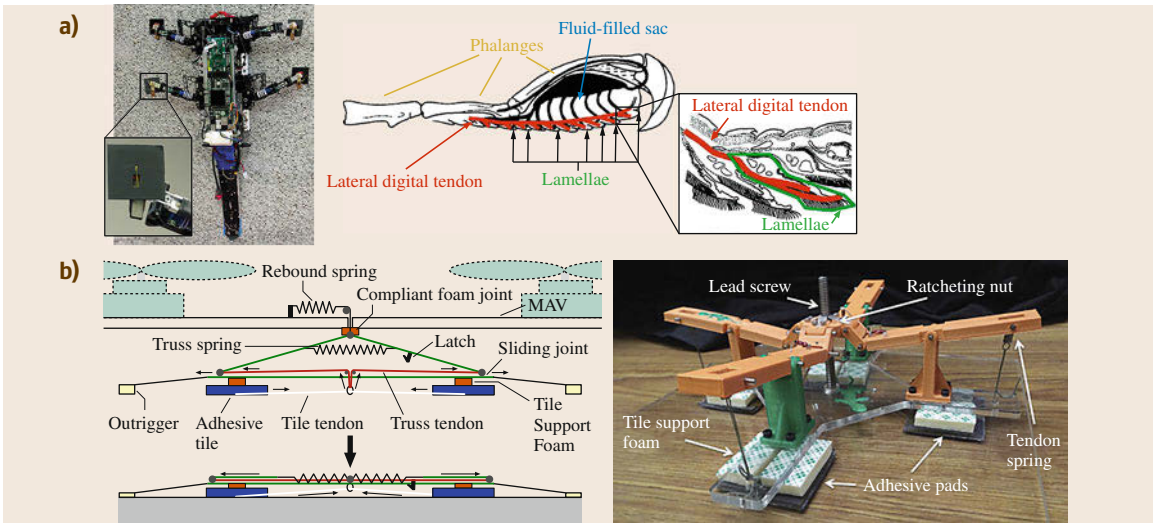


Fig.23.23a,b Graspers based on Gecko-inspired adhesives. **(a)** Gecko-inspired toe used in RiSE robot (*left*) and cross section of the gecko foot (*right*) (after [23.103]), **(b)** Collapsing truss grasper (*left*) and pivot linkage grasper (*right*) (after [23.104])

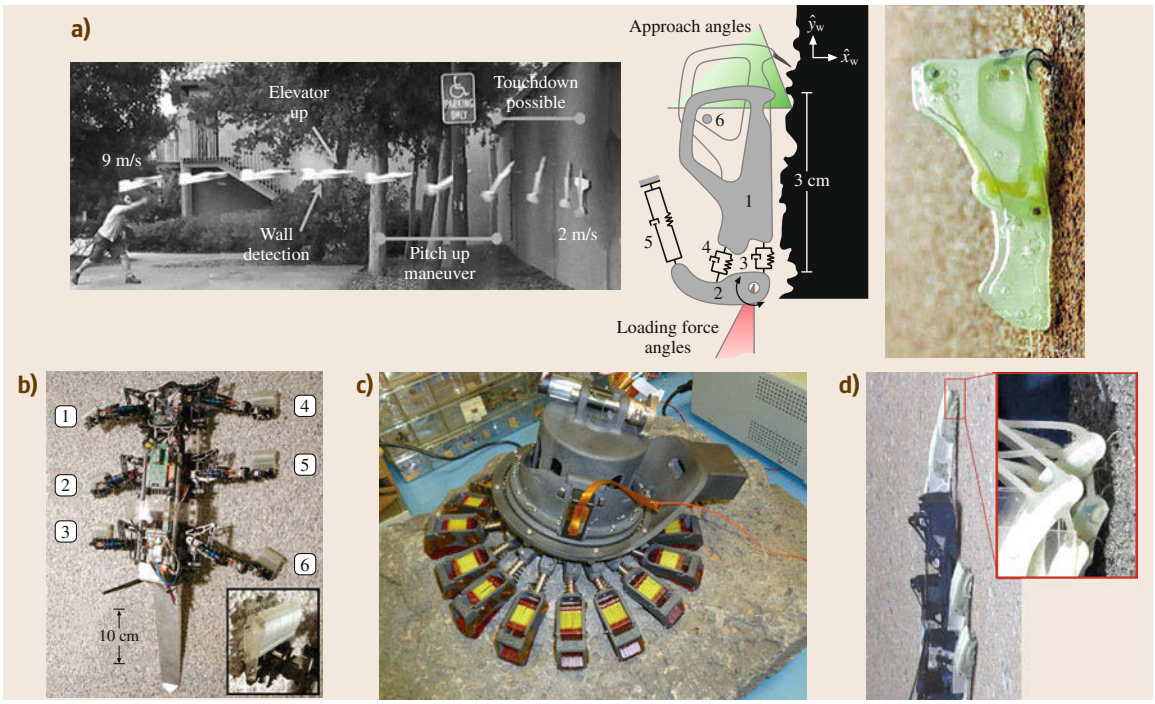


Fig.23.24a–d Microspine-based robots. **(a)** Landing and perching UAV (after [23.105]), **(b)** RiSE robot (after [23.56]), **(c)** sample acquisition tool (after [23.106]), **(d)** Spinybot (after [23.55])

tractor muscles to release the grip, which means that they do not consume any energy during gripping. Like the caterpillar, Trimmer et al. designed the grippers so that gripping is released when the SMA spring actuator is activated. *Jung* et al. proposed an underactuated mechanism based on flexural buckling [23.108]. The flexural buckling mechanism is inspired by the soft cuticle of a caterpillar's feet, which largely deforms depending on the shape of the contacting surface. The large deformation in the engineered device, flexural

Avidbots Ex1016
Page 599 of 2154

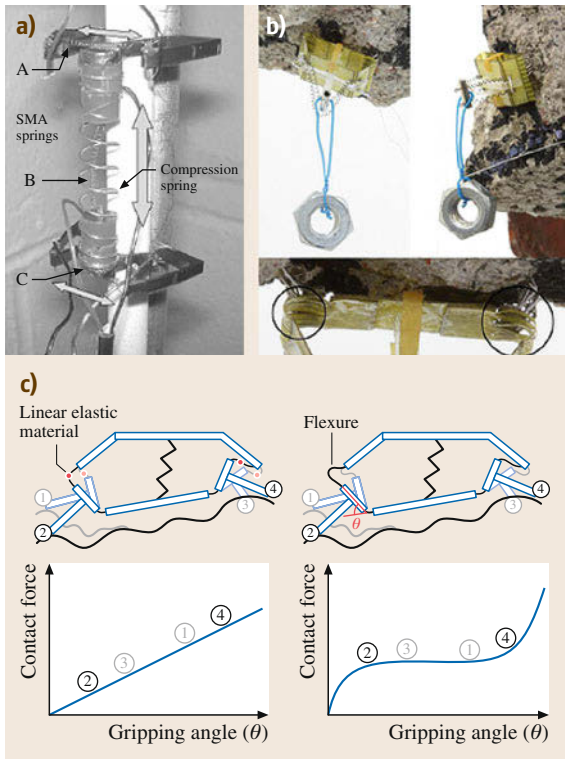


Fig. 23.25 **(a)** Passive gripping system (after [23.107]), **(b)** caterpillar-inspired underactuated gripper and **(c)** constant force region by flexural buckling (after [23.108]) ◄

buckling with an adequately selected length, provides wide gripping range with a narrow range of force variation. This provides a sufficient number of contacts with even contact forces, enabling adaptive gripping on various surfaces. In addition, design of the gripper can easily be scaled up or down depending on required scale. VIDEO 409 shows the small and large scale gripper that can achieve adaptive grasping (Fig. 23.25).

The octopus performs crawling movements with the same limbs used for grasping and manipulation, as shown in VIDEO 411. *Calisti* et al. proposed an octopus-inspired solution for movement and manipulation [23.109]. To implement octopus-like motion, they employed a steel cable for elongating and shortening and fiber cables for bending, which is inspired by the longitudinal muscles found in an octopus shown in Fig. 23.26.

Kim et al. developed flytrap-inspired high-speed gripper, as shown in VIDEO 410 [23.110]. Flytraps achieve fast capturing by using the bistable structural characteristic of its leaf. To achieve similar bistability, Kim et al. used asymmetrically laminated carbon fiber reinforced prepregs (CFRP). They also utilized a developable surface having kinematic constraints, which constrain the curvature of the artificial leaf. Therefore, the curved leaf can be actuated by bending the straight edge orthogonal to the curve, a process called bending propagation.

Doyle et al. developed an avian-inspired passive perching mechanism for quadrotors for perch-and-stare, as shown in VIDEO 415 [23.111]. Songbirds had evolved to sleep while perching. When they perch on a branch, the tendon connected from the ankle and

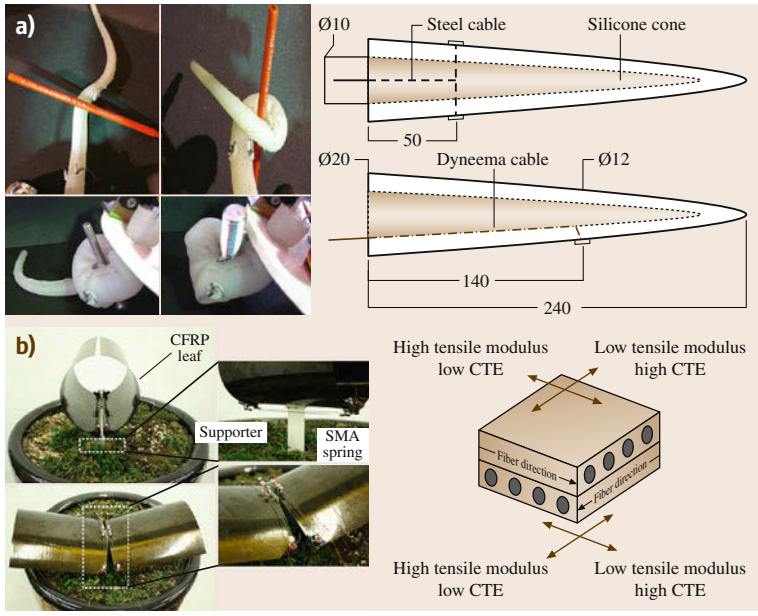


Fig. 23.26 **(a)** Octopus-inspired manipulation (*left*) and tendon-driven mechanism (*right*) (after [23.109]), **(b)** Flytrap-inspired gripper (*left*) and Orthogonally laminated CFRP (*right*) (after [23.110])

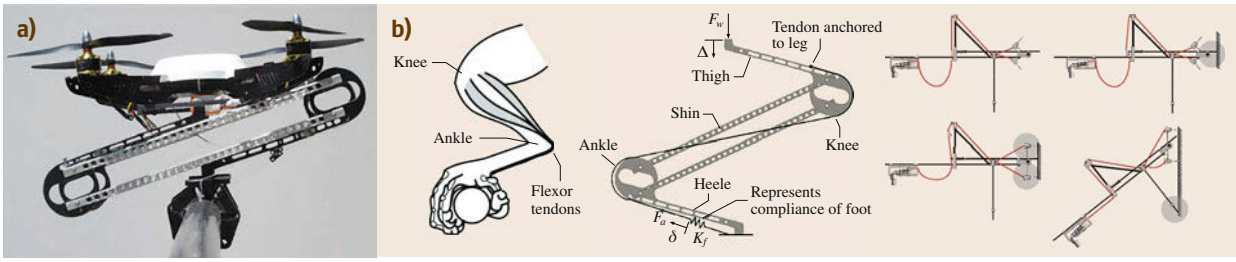


Fig. 23.27 **(a)** Avian-inspired perching mechanism with UAV (*left*) anatomy and mechanism design (*right*) (after [23.111]), **(b)** perching mechanism and process (after [23.112])

the rear side of toe automatically cause the toe to grip the branch. This allows the songbirds to tightly grip the branch without any muscular effort. Inspired by this, Doyle et al. used a four-bar mechanism and a tendon connected from the knee to the ankle and foot to couple landing motion with grasping. *Kovač* et al. presented a 4.6 g perching mechanism for microaerial vehicles (MAVs) to make them perch on various walls such as tree and concrete buildings [23.112], as shown in VIDEO 416. To achieve high impact force, the needles snap through as the trigger collides with the target surface (Fig. 23.27).

23.4 Material and Fabrication

23.4.1 Shape Deposition Manufacturing

The fundamental concept of shape deposition manufacturing (SDM) is layered molding manufacturing with CNC machining process. It not only create complex 3-D shapes rapidly, but also enables high precision finishing and large design flexibility. This concept is initially proposed by *Weiss* et al. [23.113]. Figure 23.28a shows the steps of the SDM process and Fig. 23.28b shows fabrication result [23.114]. After deposing support material, the support was fabricated by CNC machining to make high precision surface. *Li* et al. shows that it is possible to embed a various functional material such as sensors to the structure [23.115] and *Marra* et al. show that this process can be used in the fabrication of scaffolds for bone tissue engineering [23.116].

In 1999, the SDM process was first used for robot design by *Bailey* et al. [23.114]. Robot design methods based on the SDM process have advantages that it does not need complex assembly or connecting methods, and it can embed sensors and actuators directly into the body structure. In other words, the structure is build and assembled at the same time, and this

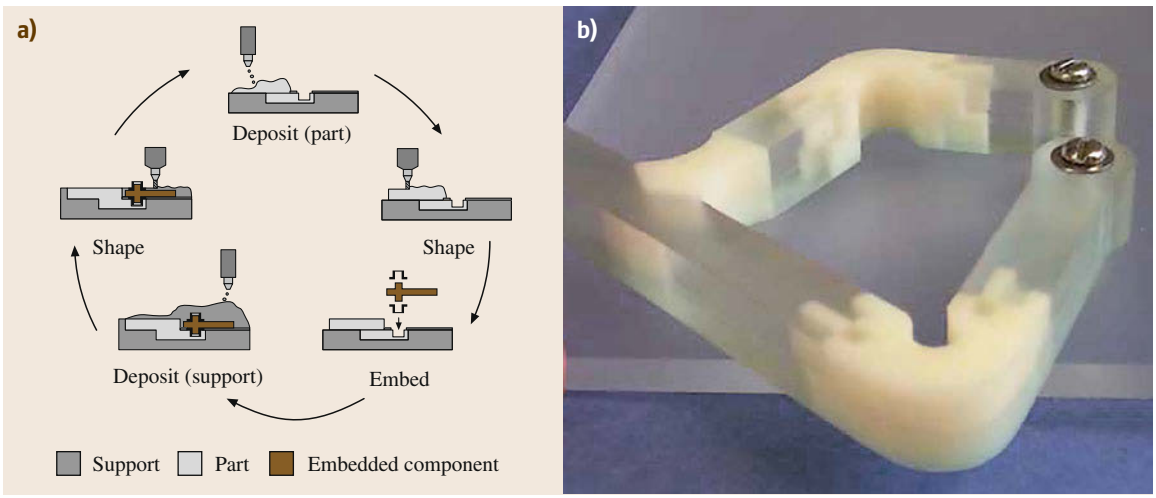


Fig.23.28a,b Shape deposition manufacturing **(a)** Process, **(b)** result of SDM process for robot mechanism design (after [23.114])

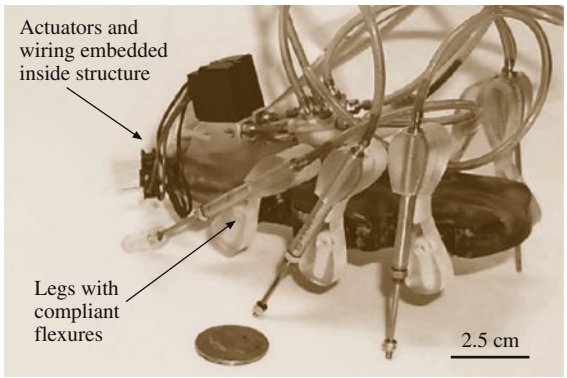


Fig. 23.29 Hexapedal robot built by SDM process (after [23.117])

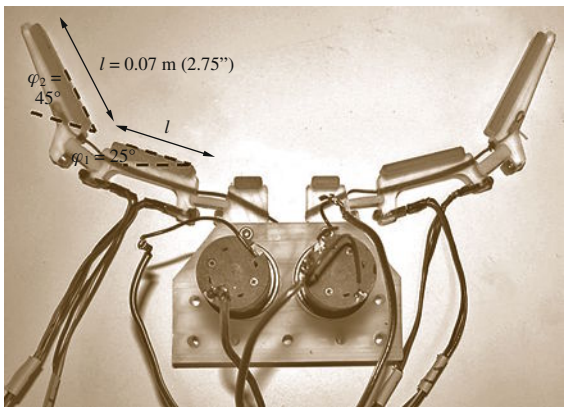


Fig. 23.30 Robotic gripper built by SDM process (after [23.118])

characteristic leads to enhanced manufacturability in small-scale applications. To build a desired mechanism, the author used multiple cycles of material deposition and shaping as shown in Fig. 23.28a. Multiple materials were used in fabrication to create variable characteristic to each functional part. More flexible material was used for joints and rigid material was used for links. The flexible components are not only used as articulated joints, but also as dampers and springs to control the impedance of a mechanism. *Binnard* et al. suggested a design framework for the SDM process for mechanism design, making more multifarious applications possible [23.119]. Figure 23.29 shows Sprawlita, a hexapedal robot fabricated based on the SDM process. Using the SDM process, actuators and wires are embedded in the body structure, resulting in robust performance and minimal manual assembly operations [23.117]. Robotic grippers can also be made by the SDM process (Fig. 23.30) [23.118]. All of joints, links, and sheath for the actuating wire were fabricated and assembled simultaneously. Embedded sensors are also possible. Force sensors embedded in the fingertips of a robotic SDM gripper have been demonstrated [23.120]. Using a soft base material for the robot, it is also possible to design a human-friendly SDM robot [23.121].

23.4.2 Smart Composite Microstructures

In the late 1990s, researchers at the University of California, Berkeley began a project to create a robotic insect capable of sustained autonomous flight – the micromechanical flying insect was born [23.122]. Among the many challenges for this project, how to manufacture and what materials should be used for structures, mechanisms, and actuators were primary concerns. Recognizing the lack of a viable *meso-scale* manufacturing method, the team lead by Fearing attempted multiple techniques including folded triangular stainless steel beams [23.123] and eventually settled on multilayer composites [23.124]. In this paradigm, later called smart composite microstructures (SCM) [23.50], layers of materials are machined, aligned, and laminated to form a quasi-two-dimensional (quasi-2-D) laminate. The choice of materials, 2-D layer geometries, and order of the layup allows the user to create an array of rigid components separated by compliant flexures. This composite laminate, later called a *standard linkage layer* can then be folded into 3-D shapes and mechanisms.

The SCM process presents a new paradigm of design and fabrication for developing small-scale robots. Planar fiber-reinforced prepreg (FRP) and flexure hinges replace conventional links and joints in the robot mechanism. The composite laminating process is

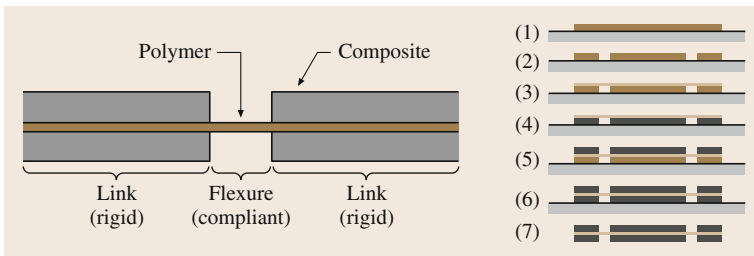


Fig. 23.31 Schematic illustration of a single joint unit in SCM and the layup of laminating process (after [23.50])

Avidbots Ex1016
Page 602 of 2154

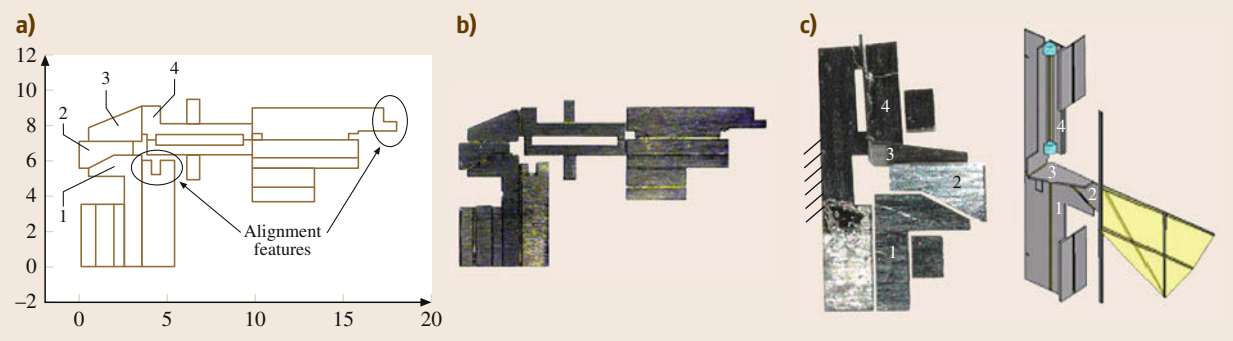


Fig.23.32a–c Spherical five-bar linkage structure created by the SCM fabrication process (after [23.52]). **(a)** The pattern design of a rigid face sheet, **(b)** laser-cut and cured microcomposite sheet before folding, **(c)** SCM spherical five-bar linkage for transmission of MFI

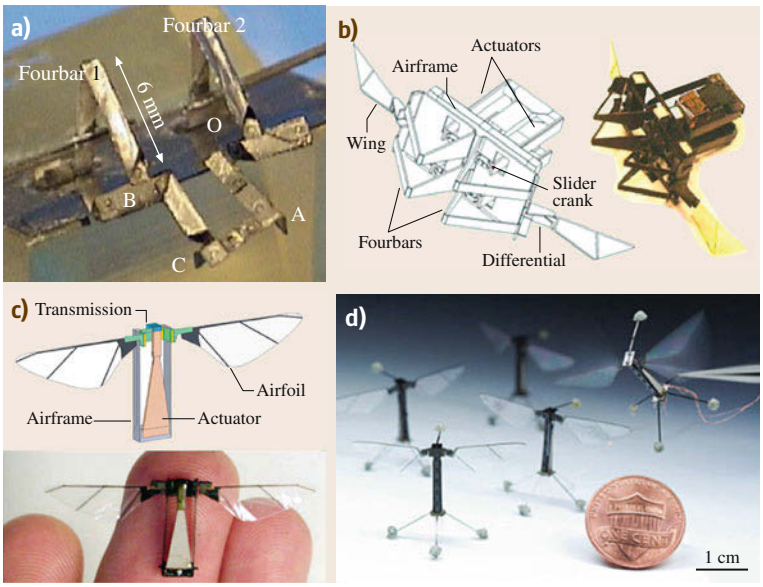


Fig.23.33a–d Various generations of MFI. **(a)** Wing transmission made of steed face sheet (after [23.123]). **(b)** Thorax mechanism with five-bar spherical linkage SCM (after [23.124]). **(c)** The Harvard robotic fly is capable of flapping-wing liftoff (after [23.50]). **(d)** The Harvard robotic fly is the first robotic insect capable of unconstrained controlled flight (after [23.125])

adopted instead of conventional machining and assembly processes which are difficult to apply to building a small-scale robot mechanism. Figure 23.31 shows a single unit of the links and the flexure hinge joints which are the key building blocks of the SCM process [23.50]. The flexure joint is a polymer film that can be bent easily and eliminating friction losses that are the dominant cause of reduced efficiency in small-scale robot mechanisms.

Rigid face sheets of fiber-reinforced composites sandwich polymers and joints are created at the gap between face sheets. The resulting quasi-2-D sheets are then folded into 3-D structures. The face sheets of rigid composite materials are carefully patterned to create robot mechanisms. For example, Fig. 23.32a is a 2-D pattern designed for creating a spherical five-bar linkages as shown in Fig. 23.32c. This is the wing transmission linkage in the early versions of micromechanical flying insect (MFI) [23.52].

Figure 23.33 shows the various versions of the MFI developed using the SCM fabrication process. Figure 23.33a is the early version of a transmission that uses a steel face sheet [23.123]. Figure 23.33b is the next generation of the MFI fabricated using a carbon fiber composite (CF) sheet and polyester film. It has 26 joints, 4-DOF, four actuators, and two wings. CF composites materials improve the performance of MFI thorax structure by reducing the inertia by a factor of three and increasing the resonant frequency by 20% [23.124]. A later version with a 3-DOF transmission (two passive DOF) and a single bimorph PZT actuator as shown in Fig. 23.33c was able to produce sufficient thrust to achieve liftoff [23.52]. Passive dynamics in the thorax structures simpli-

Avidbots Ex1016
Page 603 of 2154

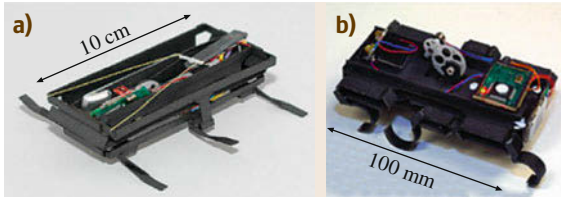


Fig. 23.34 DASH (after [23.8])

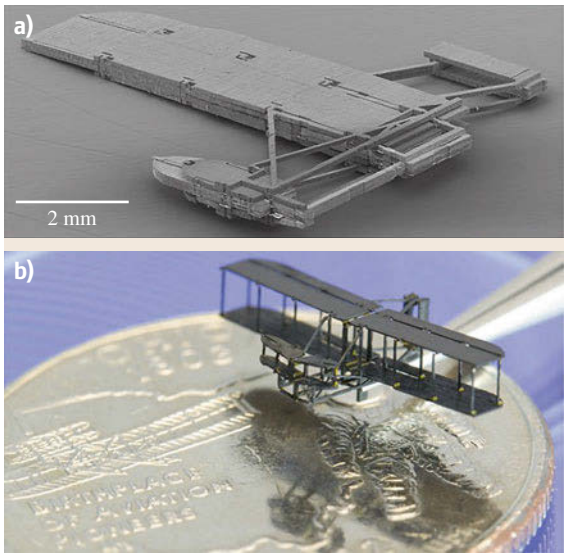


Fig.23.35a,b 1:900 scale 1903 Wright Flyer model 14 mm in wing span (after [23.126]). After laminating layup **(a)**, model after folding **(b)**

fies the design and reduces undesired coupling between the degrees of freedom. Controlled flight is accomplished by separated actuator design shown in Fig. 23.33d. Tethered but unconstrained stable hovering and basic controlled manoeuvres are demonstrated [23.125].

The SCM fabrication process can be extended to larger scales by using various sheet materials. Figure 23.34 shows centimetre scale crawling robots that are built using the SCM fabrication process with cardboard and adhesive films in place of the composites used in the MFI. This new paradigm is fast and inexpensive – both in the materials used and the required infrastructure. Furthermore, novel bio-inspired robot mechanism that produce high performance can be created easily as shown in Fig. 23.34. Figure 23.34a is a small, lightweight, power autonomous running robot, DASH [23.8]. It is capable of running at speeds up to 15 body lengths per second and surviving falls from large heights, due to the unique compliant nature of its structures. The design can be modified easily and achieve high performance with regard to stability, speed, and maneuverability as shown in Fig. 23.34b [23.10].

23.4.3 Pop-Up Book MEMS

In order to streamline the development of high performance and economical robots, many assembly tools have been developed to assist with robot construction at various scales. However, in millimetre-scale robots, many challenges arise in the fabrication process and handling many individual parts for assembly. Pop-up books and paper folding inspire the solution for eliminating the onerous assembly process with small individual parts in the fabrication process of the millimetre-scale robotic structures. Monolithic fabrication using pop-up book-inspired designs enables efficient batch processing starting from multiple layered composites similar to the basic elements used in SCM. The carefully designed layers are interconnected and allow folding mechanisms of high complexity. Once folding ensues, the complex 3-D structures are created by parallel mechanisms created in the thickness

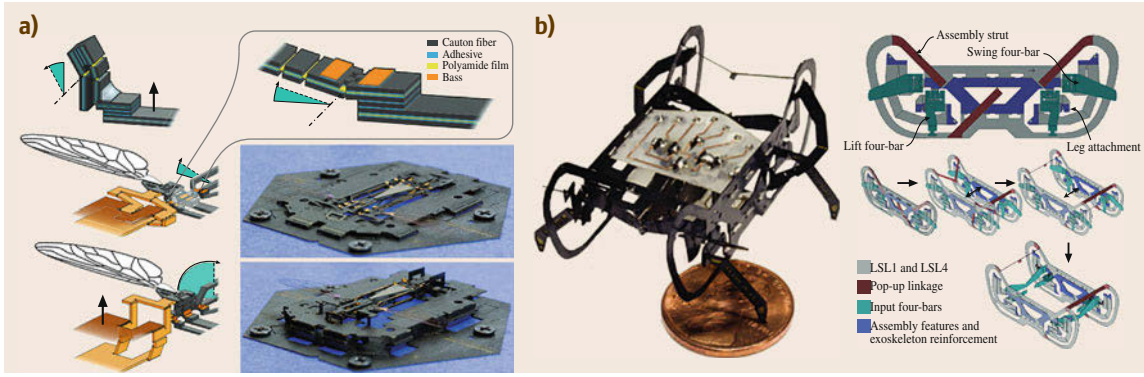


Fig. 23.36 **(a)** Mobee, monolithic design of MFI [23.127], **(b)** HAMR-VP, a 1.27 g quadrupedal microrobot manufactured using the PC-MEMS and pop-up assembly techniques (after [23.16])

Avidbots Ex1016
Page 604 of 2154

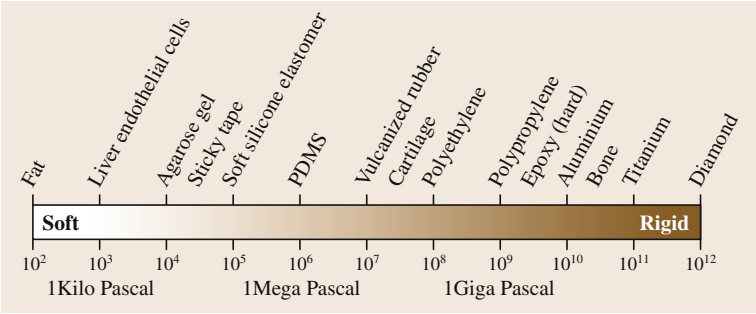


Fig. 23.37 Young's modulus for various materials (after [23.128] which was also adapted by *Autumn* et al. in [23.63])

of the laminate. Figure 23.35 presents a pop-up structure of *1:900 scale 1903 Wright Flyer model*, 14 mm in wing span [23.126]. The model consists of six rigid CF composite layers, seven adhesive layers, and two polymer flexure layers. Multiple rigid-flex folding layers are stacked and selectively bonded. The idea is previously applied to microelectric mechanical systems (MEMS) process. Combining this idea and pop-up book designs, the fabrication process is developed for building micro robotic systems including robot structures and actuators.

Figure 23.36 shows two examples that are designed and fabricated by the pop-up book MEMS process. Figure 23.36a is the Mobee (Harvard robotic fly using the monolithic popup book MEMS design methods) [23.127]. In VIDEO 398, it demonstrates how the pop-up book MEMS process enables mass production by parallel manufacturing on a single sheet and reduces entire fabrication time by eliminating onerous assembly tasks. Figure 23.36b is a small-scale quadruped crawling robot designed by pop-up book MEMS techniques. This device demonstrates how complex millimeter-scale robot structures are capable of pop-up assembly [23.16]. Twenty-three material layers are cut by precision laser machining and laminated with selective adhesion. After popup, the body frame is created and other components such as a circuit board and actuators are bonded on the frame.

23.4.4 Other Fabrication Methods

In nature, animals use soft parts of their body for generating locomotion, morphing configuration, and adapting to the environment. In order to maximize the utility of their compliance, some bio-inspired robots are primarily composed of soft materials such as fluids, gels, granules, and soft polymers. With respect to compliance, soft materials vary with a wide range of elastic (Young's) modulus [23.128] (Fig. 23.37). Therefore, bio-inspired robots should be built by different fabrication methods depending on their constituent materials.

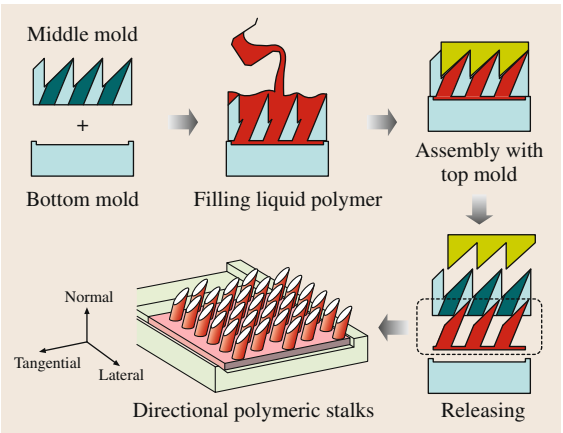


Fig. 23.38 Manufacturing process of directional adhesion pad (after [23.58])

Soft Lithography

Soft lithography was originally proposed for micro- and nanostructure manufacturing in 1998 [23.129]. Early soft lithography used an elastomeric stamp with patterned relief structures on its surface to generate desired patterns and structures. In bio-inspired robotics, for example, a directional adhesion pad inspired by a gecko foot's microscale adhesion spines was developed by a similar fabrication method to soft lithography [23.58] (Fig. 23.38).

Advances of convenient, effective and low-cost soft lithography allowed it to be utilized not only in the micro and nano scale in bio-chemical fields, but also in macro-scale robotics. Moreover, development and popularization of additive manufacturing process using 3-D printers which can design molds with various shapes easily and rapidly have helped soft lithography to be applied in robotics. Soft lithography has become a representative manufacturing method for bio-inspired soft robots since soft lithography was first applied to manufacturing bio-inspired soft robots [23.130–133] (Fig. 23.39). In soft lithography for bio-inspired soft robots, uncured elastomeric polymers, such as PDMS or EcoFlex, are poured into a mold designed with

Avidbots Ex1016
Page 605 of 2154

configuration of the structure. After curing, the soft materials form a structure containing multiple air chambers and pneumatic channels. Such kind of structure is so-called a pneumatic network (PneuNet) or bending fluidic actuator (BFA).

This structure allows the robots to generate sophisticated locomotion such as gripping, walking, and crawling as shown in Fig. 23.40. In addition, extra embedded channels can control the flow of a dyeing solution so that a soft robot can camouflage by changing body color to match the color of the surroundings [23.135]. The actuation or locomotion of robots manufactured by soft lithography depends on the elastic modulus of materials and the geometry and position of air chambers [23.136].

Furthermore, soft lithography in robotics has advanced by mixing different kind of materials. By embedding sheets or fibers into elastomers, an actuator has asymmetric compliance which allows the structure to be flexible but not extensible, so that the actuator can generate a wide range of motions such as bending, extension, contraction, twisting, and others [23.137] (Fig. 23.41). By embedding magnets into elastomers, a soft robot can attach, detach, and easily align modules that have a unique function per each module depending on the task [23.138].

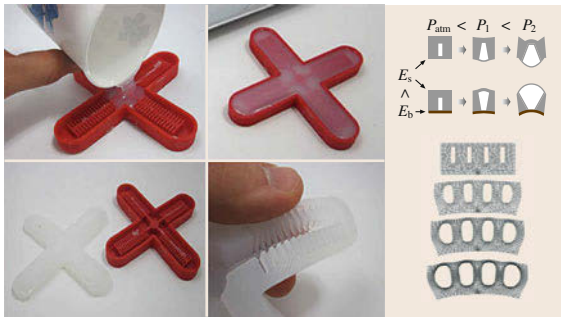


Fig. 23.39 Soft lithography of pneumatic network (PneuNet) or bending fluidic actuator (BFA) (after [23.130, 133, 134])



Fig. 23.40 Bio-inspired soft robots and actuators manufactured by soft lithography (after [23.130, 131, 135, 136])

To overcome a slow actuation, which is a typical issue for soft robots, the design of segmented air chambers divided by slits was introduced as an alternative soft actuator [23.139]. As a result, a fast PneuNet structure had high rate of actuation, improved 25 times relative to the slow PneuNet actuators. Also, a reduced change of volume minimizes fatiguing the materials, and thus the durability improves to a level that the actuator does not fail within a million cycles of full bending.

Actuator Embedded Molding

Bio-inspired soft robots potentially have infinite degrees of freedom and the nonlinearity of soft materials creates difficulty in generating desired postures and motion. Actuator-embedded molding is commonly used to manufacture the soft structure for bio-inspired robots. In actuator-embedded molding, the design considerations are the type of the actuator and the actuator's location and direction in the soft structure. For an example of a bio-inspired robot using actuator-embedded molding, a turtle-like swimming robot was created using a smart soft composite (SSC) structure to create bending and twisting deformation [23.77] (Fig. 23.42). The SSC structure consists of an actuator-embedded layer as an active component, a patterned layer as a passive component, and a soft matrix as the body as shown in Fig. 23.42. The angle of the patterned layer determines the bending direction passively, and the actuator-embedded layer generates deformation. The soft matrix helps to deform the structure continuously. Widely used

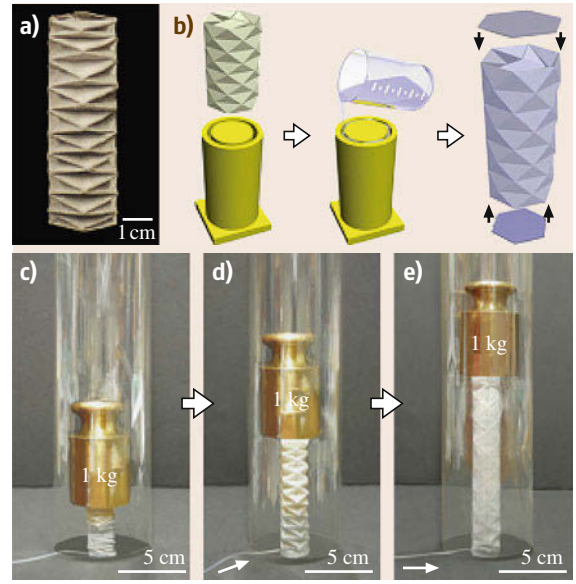


Fig.23.41a–e Programmable paper–elastomer composites-based pneumatic actuators by using soft lithography (after [23.137])

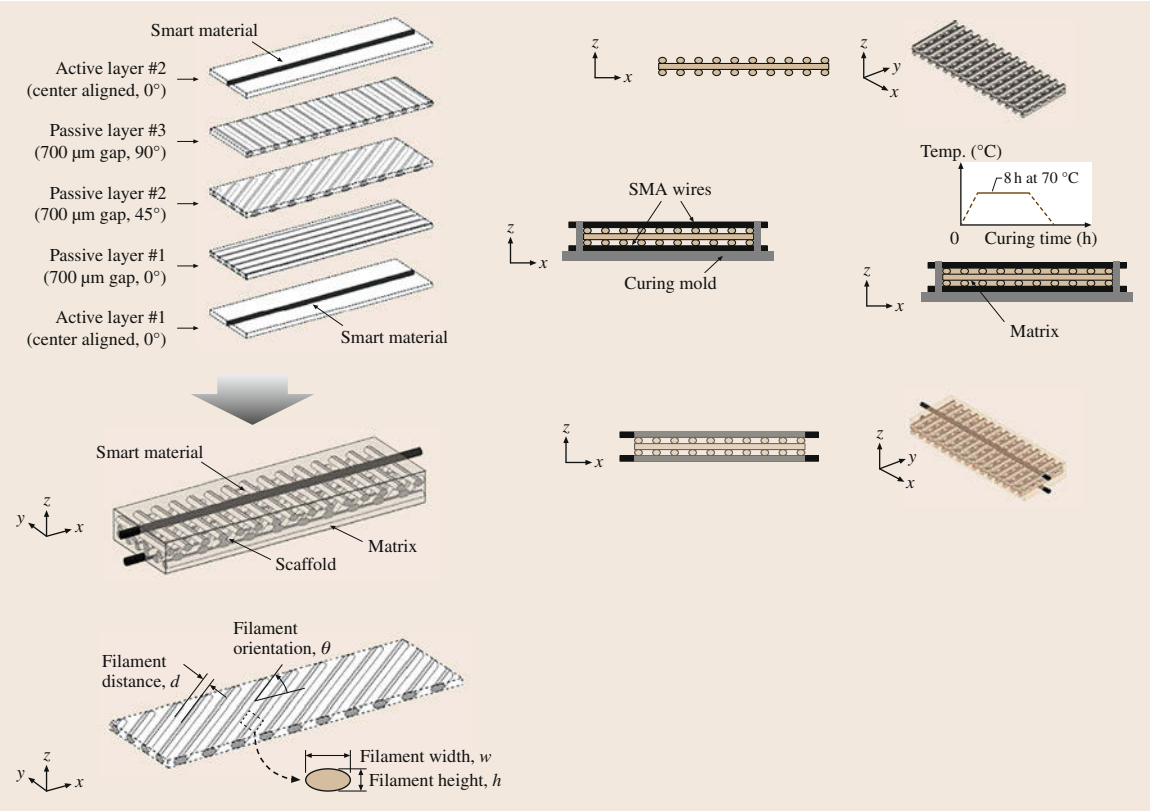


Fig. 23.42 Actuator-embedded molding for the SSC structure actuator (after [23.77])

actuators in the actuator embedded molding are wires including common wires connected with servo motors and shape memory alloys (SMA) [23.25, 77, 140].

Additive Manufacturing

Additive manufacturing or 3-D printing is a rapid prototyping (RP) process of manufacturing a 3-D solid structure of any configuration through sequential layering from a digital CAD model. Since the 1980s, various types of additive manufacturing have been developed: solid-based processes such as fused deposition modeling (FDM), photo curable liquid polymer-based processes such as stereolithography (SLA), and powder-based processes such as selective laser sintering (SLS).

Evolving the technology of 3-D printers, soft material deposition and even hybrid deposition of different materials with different stiffnesses are available. Thus, soft structures can be built at once, and products using hybrid deposition can have features of rigidity and softness at the same time. For example, a highly deformable 3-D printed soft robot was developed [23.141]. This robot body was printed by using a multimaterial printable 3-D printer (Objet Connex 500TM 3-D printer) with two materials: one is a soft rubber-like Objet TangoBlackPlusTM and another is a hard polypropylene-like Objet VeroWhitePlusTM. These two materials have different friction coefficients; thus, the robot can switch friction with the ground on its edge by bending the body.

23.5 Conclusion

Biomimetic robotics attempt to create devices that are capable of various types of effective interaction with natural environment, e.g., locomotion and manipulation, by using the principles of nature. Nature is full of surprising types of movements that enables insects and animals to survive by escaping from danger or hunting for food. In this chapter, we have presented robots that attempt to recreate these feats by understanding the un-

Avidbots Ex1016
Page 607 of 2154

derlying principles, deriving an engineering design, and fabricating them with novel methods. In recent years, engineers have succeeded in recreating insects and animals that show amazing capabilities such as climbing walls like a gecko, hovering like a fly and climbing trees like a snake. These robots have been developed to understand nature, and also to be used as tools for surveillance, information gathering, and rescue operations. However, many of these robots are still in the basic research phase and are not yet ready for everyday use.

The key open research issues remain in the broad areas of materials, fabrication, actuation, and power. Composites and polymers, together with novel fabrication methods, have enabled various novel biomimetic robots. Development of these new material and fabrication has been one of the key enabling technologies, and further development of these technologies will certainly contribute to more mature biomimetic robots. Actuation and power still remain the bottleneck of many biomimetic robots. DC motors are the actuators of choice for many biomimetic robots, and with novel transmission design, DC motors can create motions required by the robots. However, DC motors are inefficient for small-scale biomimetic robots. Although artificial muscle actuators that are based on shrinkage or expansion of material, e.g., shape memory alloys, IPMC, electro-active polymers and shape memory polymers, promise to give robots capabilities similar to creatures that use biological muscles even at small scales. However, many issues related to robustness, efficiency, and power limit the capabilities of robots that use these actuators. Limitations of these actuators should be carefully considered to match the desired application. Development of new artificial muscle actuator that can emulate biological muscles – without the drawbacks of current actuators – is needed to open up a new era for biomimetic robots. Batteries also limit the capabilities of current biomimetic robots compared to their biological counterparts in terms of size and operation time. Development of energy-harvesting technologies together with new battery chemistries and manufacturing methods will enable longer operation time, which will contribute to realizing wider applications for biomimetic robots. Overall, biomimetic robot design is one of the most challenging areas of robot design since it requires development of various technologies to mimic the structure and function of natural systems. Therefore, development of biomimetic robots can have broader impact in many areas of engineering and science, and should be seen as a platform for various convergent technologies.

Video-References

VIDEO 278 The long-jumping robot 'Grillo'
available from http://handbookofrobotics.org/view-chapter/23/videodetails/278

VIDEO 279 A miniature 7 g jumping robot
available from http://handbookofrobotics.org/view-chapter/23/videodetails/279

VIDEO 280 A single motor actuated miniature steerable jumping robot
available from http://handbookofrobotics.org/view-chapter/23/videodetails/280

VIDEO 281 The Flea: Flea-inpired light jumping robot using elastic catapult with active storage and release mechanism
available from http://handbookofrobotics.org/view-chapter/23/videodetails/281

VIDEO 285 Jumping & landing robot 'MOWGLI'
available from http://handbookofrobotics.org/view-chapter/23/videodetails/285

VIDEO 286 RoACH: A 2.4 g, untethered crawling hexapod robot
available from http://handbookofrobotics.org/view-chapter/23/videodetails/286

VIDEO 287 A new form of peristaltic locomotion in a robot
available from http://handbookofrobotics.org/view-chapter/23/videodetails/287

VIDEO 288 Meshworm
available from http://handbookofrobotics.org/view-chapter/23/videodetails/288

VIDEO 289 Treebot: Autonomous tree climbing by tactile sensing
available from http://handbookofrobotics.org/view-chapter/23/videodetails/289

VIDEO 290 Omegabot : Inchworm inspired robot climbing
available from http://handbookofrobotics.org/view-chapter/23/videodetails/290

VIDEO 291 GoQBot: Insanely fast robot caterpillar
available from http://handbookofrobotics.org/view-chapter/23/videodetails/291

VIDEO 388 SpinybotII: Climbing hard walls with compliant microspines
available from http://handbookofrobotics.org/view-chapter/23/videodetails/388

VIDEO 389 Smooth vertical surface climbing with directional adhesion
available from http://handbookofrobotics.org/view-chapter/23/videodetails/389

Avidbots Ex1016
Page 608 of 2154

VIDEO 390 Biologically inspired climbing with a hexapedal robot
available from http://handbookofrobotics.org/view-chapter/23/videodetails/390
VIDEO 391 CLASH: Climbing vertical loose cloth
available from http://handbookofrobotics.org/view-chapter/23/videodetails/391
VIDEO 392 Torque control strategies for snake robots
available from http://handbookofrobotics.org/view-chapter/23/videodetails/392
VIDEO 393 Snake robot climbs a tree
available from http://handbookofrobotics.org/view-chapter/23/videodetails/393
VIDEO 394 Snake robot in the water
available from http://handbookofrobotics.org/view-chapter/23/videodetails/394
VIDEO 395 Salamandra Robotica II robot walking and swimming
available from http://handbookofrobotics.org/view-chapter/23/videodetails/395
VIDEO 397 ACM-R5H
available from http://handbookofrobotics.org/view-chapter/23/videodetails/397
VIDEO 398 Pop-up fabrication of the Harvard monolithic bee (Mobee)
available from http://handbookofrobotics.org/view-chapter/23/videodetails/398
VIDEO 399 Controlled flight of a biologically-inspired, insect-scale robot
available from http://handbookofrobotics.org/view-chapter/23/videodetails/399
VIDEO 400 Rhex the parkour robot
available from http://handbookofrobotics.org/view-chapter/23/videodetails/400
VIDEO 401 Mini whegs
available from http://handbookofrobotics.org/view-chapter/23/videodetails/401
VIDEO 402 Robot dragonfly DelFly explorer flies autonomously
available from http://handbookofrobotics.org/view-chapter/23/videodetails/402
VIDEO 403 Standford Sprawl and iSprawl
available from http://handbookofrobotics.org/view-chapter/23/videodetails/403
VIDEO 405 DASH: Resilient high-speed 16 g hexapedal robot
available from http://handbookofrobotics.org/view-chapter/23/videodetails/405
VIDEO 406 HAMR3: An autonomous 1.7 g ambulatory robot
available from http://handbookofrobotics.org/view-chapter/23/videodetails/406
VIDEO 407 Undulatory gaits in a centipede millirobot
available from http://handbookofrobotics.org/view-chapter/23/videodetails/407
VIDEO 408 VelociRoACH
available from http://handbookofrobotics.org/view-chapter/23/videodetails/408
VIDEO 409 Underactuated adaptive gripper using flexural buckling
available from http://handbookofrobotics.org/view-chapter/23/videodetails/409
VIDEO 410 Flytrap-inspired bi-stable gripper
available from http://handbookofrobotics.org/view-chapter/23/videodetails/410
VIDEO 411 An octopus-bioinspired solution to movement and manipulation for soft robots
available from http://handbookofrobotics.org/view-chapter/23/videodetails/411
VIDEO 412 Landing and perching UAV
available from http://handbookofrobotics.org/view-chapter/23/videodetails/412
VIDEO 413 Dynamic surface grasping with directional adhesion
available from http://handbookofrobotics.org/view-chapter/23/videodetails/413
VIDEO 414 Gravity-independent rock-climbing robot and a sample acquisition tool with microspine grippers
available from http://handbookofrobotics.org/view-chapter/23/videodetails/414
VIDEO 415 Avian-inspired perching mechanism with UAV
available from http://handbookofrobotics.org/view-chapter/23/videodetails/415
VIDEO 416 A perching mechanism for micro aerial vehicles
available from http://handbookofrobotics.org/view-chapter/23/videodetails/416
VIDEO 431 G9 series robotic fish
available from http://handbookofrobotics.org/view-chapter/23/videodetails/431
VIDEO 432 Ichthus
available from http://handbookofrobotics.org/view-chapter/23/videodetails/432
VIDEO 433 Autonomous, self-contained soft robotic fish
available from http://handbookofrobotics.org/view-chapter/23/videodetails/433
VIDEO 434 Robotic Ray takes a swim
available from http://handbookofrobotics.org/view-chapter/23/videodetails/434

Avidbots Ex1016
Page 609 of 2154

References

23.1 R.J. Full, K. Autumn, J. Chung, A. Ahn: Rapid negotiation of rough terrain by the death-head cockroach, Am. Zool. **38**(5), 81A (1998)

23.2 R.J. Full, M.S. Tu: Mechanics of a rapid running insect: Two-, four- and six-legged locomotion, J. Exp. Biol. **156**, 215–231 (1991)

23.3 C.P. Ellington: The novel aerodynamics of insect flight: Applications to micro-air vehicles, J. Exp. Biol. **202**, 3439–3448 (1999)

23.4 U. Saranli, M. Buehler, D.E. Koditschek: RHex: A simple and highly mobile hexapod robot, Int. J. Robotics Res. **20**, 616–631 (2001)

23.5 J.M. Morrey, B. Lambrecht, A.D. Horchler, R.E. Ritzmann, R.D. Quinn: Highly mobile and robust small quadruped robots, Proc IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS), Vol. 1 (2003) pp. 82–87

23.6 J.E. Clark, J.G. Cham, S.A. Bailey, E.M. Froehlich, P.K. Nahata, R.J. Full, M.R. Cutkosky: Biomimetic design and fabrication of a hexapedal running robot, Proc. IEEE Int. Conf. Robotics Autom. (ICRA), Vol. 4 (2001) pp. 3643–3649

23.7 S. Kim, J.E. Clark, M.R. Cutkosky: iSprawl: design and tuning for high-speed autonomous open-loop running, Int. J. Robotics Res. **25**, 903–912 (2006)

23.8 P. Birkmeyer, K. Peterson, R.S. Fearing: DASH: A dynamic 16g hexapedal robot, Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS) (2009) pp. 2683–2689

23.9 A.M. Hoover, E. Steltz, R.S. Fearing: RoACH: An autonomous 2.4g crawling hexapod robot, Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS) (2008) pp. 26–33

23.10 A.M. Hoover, S. Burden, X.-Y. Fu, S.S. Sastry, R.S. Fearing: Bio-inspired design and dynamic maneuverability of a minimally actuated six-legged robot, Proc. IEEE/RAS Biomed. Robotics Biomech. (BioRob) (2010) pp. 869–876

23.11 A.O. Pullin, N.J. Kohut, D. Zarrouk, R.S. Fearing: Dynamic turning of 13 cm robot comparing tail and differential drive, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (2012) pp. 5086–5093

23.12 A.T. Baisch, C. Heimlich, M. Karpelson, R.J. Wood: HAMR3: An autonomous 1.7g ambulatory robot, Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS) (2011) pp. 5073–5079

23.13 K.L. Hoffman, R.J. Wood: Turning gaits and optimal undulatory gaits for a modular centipede-inspired millirobot, Proc. IEEE/RAS Biomed. Robotics Biomech. (BioRob) (2012) pp. 1052–1059

23.14 R. Sahai, S. Avadhanula, R. Groff, E. Steltz, R. Wood, R.S. Fearing: Towards a 3g crawling robot through the integration of microrobot technologies, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (2006) pp. 296–302

23.15 J.T. Watson, R.E. Ritzmann, S.N. Zill, A.J. Pollack: Control of obstacle climbing in the cockroach, Blaberus discoidalis. I. Kinematics, J. Comp. Physiol. A **188**, 39–53 (2002)

23.16 A.T. Baisch, O. Ozcan, B. Goldberg, D. Ithier, R.J. Wood: High speed locomotion for a quadrupedal microrobot, Int. J. Robotics Res. **33**, 1063–1082 (2014)

23.17 D.W. Haldane, K.C. Peterson, F.L. Garcia Bermudez, R.S. Fearing: Animal-inspired design and aerodynamic stabilization of a hexapedal millirobot, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (2013) pp. 3279–3286

23.18 A.S. Boxerbaum, H.J. Chiel, R.D. Quinn: A new theory and methods for creating peristaltic motion in a robotic platform, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (2010) pp. 1221–1227

23.19 S. Seok, C.D. Onal, K.-J. Cho, R.J. Wood, D. Rus, S. Kim: Meshworm: A peristaltic soft robot with antagonistic nickel titanium coil actuators, IEEE/ASME Trans. Mechatron. **18**, 1–13 (2012)

23.20 A. Menciassi, D. Accoto, S. Gorini, P. Dario: Development of a biomimetic miniature robotic crawler, Auton. Robotics **21**, 155–163 (2006)

23.21 K. Kotay, D. Rus: The inchworm robot: A multi-functional system, Auton. Robotics **8**, 53–69 (2000)

23.22 N. Cheng, G. Ishigami, S. Hawthorne, H. Chen, M. Hansen, M. Telleria, R. Playter, K. Iagnemma: Design and analysis of a soft mobile robot composed of multiple thermally activated joints driven by a single actuator, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (2010) pp. 5207–5212

23.23 J.-S. Koh, K.-J. Cho: Omega-shaped inchworm-inspired crawling robot with large-index-and-pitch (LIP) SMA spring actuators, IEEE/ASME Trans. Mechatron. **18**, 419–429 (2013)

23.24 T.L. Lam, Y. Xu: Climbing strategy for a flexible tree climbing robot – Treebot, IEEE Trans. Robotics **27**, 1107–1117 (2011)

23.25 H.-T. Lin, G.G. Leisk, B. Trimmer: GoQBot: A caterpillar-inspired soft-bodied rolling robot, Bioinsp. Biomimet. **6**, 026007 (2011)

23.26 S. Hirose, Y. Umetani: The development of soft gripper for the versatile robot hand, Mech. Mach. Theory **13**, 351–359 (1978)

23.27 H. Ohno, S. Hirose: Design of slim slime robot and its gait of locomotion, Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS) (2001) pp. 707–715

23.28 C. Wright, A. Johnson, A. Peck, Z. McCord, A. Naaktgeboren, P. Gianfortoni, M. Gonzalez-Rivero, R. Hatton, H. Choset: Design of a modular snake robot, Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS) (2007) pp. 2609–2614

23.29 H. Yamada, S. Chigisaki, M. Mori, K. Takita, K. Ogami, S. Hirose: Development of amphibious snake-like robot ACM-R5, Proc. ISR (2005)

23.30 J. Gray: The mechanism of locomotion in snakes, J. Exp. Biol. **23**, 101–120 (1946)

23.31 G.S. Miller: The motion dynamics of snakes and worms, ACM Siggraph Comput. Graph. **22**, 169–173 (1988)

Avidbots Ex1016
Page 610 of 2154

23.32 Z. Bayraktaroglu: Snake-like locomotion: Experimentations with a biologically inspired wheelless snake robot, Mech. Mach. Theory **44**, 591–602 (2009)

23.33 D.L. Hu, J. Nirody, T. Scott, M.J. Shelley: The mechanics of slithering locomotion, Proc. Natl. Acad. Sci. **106**, 10081–10085 (2009)

23.34 Z. Wang, S. Ma, B. Li, Y. Wang: Experimental study of passive creeping for a snake-like robot, Proc. IEEE/ICME Int. Conf. Complex Med. Eng. (CME) (2011) pp. 382–387

23.35 J.J. Socha, T. O'Dempsey, M. LaBarbera: A 3-D kinematic analysis of gliding in a flying snake, Chrysopelea paradisi J. Exp. Biol. **208**, 1817–1833 (2005)

23.36 R.L. Hatton, H. Choset: Generating gaits for snake robots: Annealed chain fitting and keyframe wave extraction, Auton. Robotics **28**, 271–281 (2010)

23.37 K.J. Dowling: *Limbless Locomotion: Learning to Crawl with a Snake Robot* (NASA, Pittsburgh 1996)

23.38 S. Hirose, M. Mori: Biologically inspired snake-like robots, IEEE Int. Conf. Robotics Biomimet. (ROBIO) (2004) pp. 1–7

23.39 C. Wright, A. Buchan, B. Brown, J. Geist, M. Schwerin, D. Rollinson, M. Tesch, H. Choset: Design and architecture of the unified modular snake robot, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (2012) pp. 4347–4354

23.40 K.-H. Low: *Industrial Robotics: Programming, Simulation and Applications* (InTech, Rijeka 2007)

23.41 A.J. Ijspeert, A. Crespi, D. Ryczko, J.M. Cabelguen: From swimming to walking with a salamander robot driven by a spinal cord model, Science **315**, 1416–1420 (2007)

23.42 R. Crespi, K. Karakasiliotis, A. Guignard, A.J. Ijspeert: 1 Salamandra robotica II: An amphibious robot to study salamander-like swimming and walking gaits, IEEE Trans. Robotics **29**, 308–320 (2013)

23.43 S. Hirose, H. Yamada: Snake-like robots [Tutorial], IEEE Robotics Autom. Mag. **16**, 88–98 (2009)

23.44 N. Kamamichi, M. Yamakita, K. Asaka, Z.-W. Luo: A snake-like swimming robot using IPMC actuator/sensor, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (2006) pp. 1812–1817

23.45 P. Liljebäck, K.Y. Pettersen, O. Stavdahl, J.T. Gravdahl: Snake robot locomotion in environments with obstacles, IEEE/ASME Trans. Mechatron. **17**, 1158–1169 (2012)

23.46 P. Liljebäck, K.Y. Pettersen, Ø. Stavdahl, J.T. Gravdahl: A review on modelling, implementation, and control of snake robots, Robotics Auton. Syst. **60**, 29–40 (2012)

23.47 M.H. Dickinson: Wing rotation and the aerodynamic basis of insect flight, Science **284**, 1954–1960 (1999)

23.48 M.H. Dickinson: How animals move: An integrative view, Science **288**, 100–106 (2000)

23.49 G.C.H.E. de Croon, K.M.E. de Clercq, R. Ruijsink, B. Remes, C. de Wagter: Design, aerodynamics, and vision-based control of the DelFly, Int. J. Micro Air Veh. **1**(2), 71–97 (2009)

23.50 R.J. Wood, S. Avadhanula, R. Sahai, E. Steltz, R.S. Fearing: Microrobot design using fiber reinforced composites, J. Mech. Des. **130**, 052304 (2008)

23.51 M. Keennon, K. Klingebiel, H. Won, A. Andriukov: Development of the nano hummingbird: A tailless flapping wing micro air vehicle, AIAA Aerospace Sci. Meet. (2012)

23.52 R.J. Wood: The first takeoff of a biologically inspired at-scale robotic insect, IEEE Trans. Robotics **24**, 341–347 (2008)

23.53 K. Peterson, P. Birkmeyer, R. Dudley, R.S. Fearing: A wing-assisted running robot and implications for avian flight evolution, Bioinsp. Biomimet. **6**, 046008 (2011)

23.54 S. Hirose, A. Nagakubo, R. Toyama: Machine that can walk and climb on floors, walls and ceilings, Adv. Robotics ICAR '05. Proc. (1991) pp. 753–758

23.55 S. Kim, A.T. Asbeck, M.R. Cutkosky, W.R. Provancher: SpinybotII: Climbing hard walls with compliant microspines, Adv. Robotics ICAR '05. Proc. (2005) pp. 601–606

23.56 M. Spenko, G.C. Haynes, J. Saunders, M.R. Cutkosky, A.A. Rizzi, R.J. Full, D.E. Ko-ditschek: Biologically inspired climbing with a hexapedal robot, J. Field Robotics **25**, 223–242 (2008)

23.57 K. Autumn, A. Dittmore, D. Santos, M. Spenko, M. Cutkosky: Frictional adhesion: A new angle on gecko attachment, J. Exp. Biol. **209**, 3569–3579 (2006)

23.58 S. Kim, M. Spenko, S. Trujillo, B. Heyneman, D. Santos, M.R. Cutkosky: Smooth vertical surface climbing with directional adhesion, IEEE Trans. Robotics **24**, 65–74 (2008)

23.59 M. Minor, H. Dulimarta, G. Danghi, R. Mukherjee, R.L. Tummala, D. Aslam: Design, implementation, and evaluation of an under-actuated miniature biped climbing robot, Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS) (2000) pp. 1999–2005

23.60 D. Longo, G. Muscato: The Alicia 3 climbing robot: A three-module robot for automatic wall inspection, IEEE Robotics Autom. Mag. **13**, 42–50 (2006)

23.61 M. Armada, M. Prieto, T. Akinfiev, R. Fernández, P. González, E. García, H. Montes, S. Nabulsi, R. Ponticelli, J. Sarriá, J. Estremera, S. Ros, J. Grieco, G. Fernández: On the design and development of climbing and walking robots for the maritime industries, J. Marit. Res. **2**, 9–32 (2005)

23.62 G.C. Haynes, A. Khripin, G. Lynch, J. Amory, A. Saunders, A.A. Rizzi, D.E. Koditschek: Rapid pole climbing with a quadrupedal robot, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (2009) pp. 2767–2772

23.63 K. Autumn, Y.A. Liang, S.T. Hsieh, W. Zesch, W.P. Chan, T.W. Kenny, R. Fearing, R.J. Full: Adhesive force of a single gecko foot-hair, Nature **405**, 681–685 (2000)

23.64 G.A. Lynch, J.E. Clark, P.-C. Lin, D.E. Koditschek: A bioinspired dynamical vertical climbing robot, Int. J. Robotics Res. **31**, 974–996 (2012)

Avidbots Ex1016
Page 611 of 2154

23.65 J. Clark, D. Goldman, P.-C. Lin, G. Lynch, T. Chen, H. Komsuoglu, R.J. Full, D. Koditschek: Design of a bio-inspired dynamical vertical climbing robot, Robotics Sci. Syst. (2007)

23.66 P. Birkmeyer, A.G. Gillies, R.S. Fearing: Dynamic climbing of near-vertical smooth surfaces, Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS) (2012) pp. 286–292

23.67 K.A. Daltorio, T.E. Wei, S.N. Gorb, R.E. Ritzmann, R.D. Quinn: Passive foot design and contact area analysis for climbing mini-whegs, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (2007) pp. 1274–1279

23.68 O. Unver, A. Uneri, A. Aydemir, M. Sitti: Geckobot: A gecko inspired climbing robot using elastomer adhesives, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (2006) pp. 2329–2335

23.69 S.A. Bailey, J.G. Cham, M.R. Cutkosky, R.J. Full: A biomimetic climbing robot based on the gecko, J. Bionic Eng. **3**, 115–125 (2006)

23.70 M.P. Murphy, C. Kute, Y. Mengüç, M. Sitti: Waalbot II: Adhesion recovery and improved performance of a climbing robot using fibrillar adhesives, Int. J. Robotics Res. **30**, 118–133 (2011)

23.71 D.S. Barrett: Propulsive Efficiency of a Flexible Hull Underwater Vehicle, Ph.D. Thesis (MIT, Cambridge 1996)

23.72 J. Liu, H. Hu: Biological inspiration: From carangiform fish to multi-joint robotic fish, J. Bionic Eng. **7**, 35–48 (2010)

23.73 G.-H. Yang, K.-S. Kim, S.-H. Lee, C. Cho, Y. Ryuh: Design and control of 3-DOF robotic fish 'ICHTHUS V5, Lect. Not. Comp. Sci. **8103**, 310–319 (2013)

23.74 K. Low: Modelling and parametric study of modular undulating fin rays for fish robots, Mech. Mach. Theory **44**, 615–632 (2009)

23.75 A.D. Marchese, C.D. Onal, D. Rus: Autonomous soft robotic fish capable of escape maneuvers using fluidic elastomer actuators, Soft Robotics **1**, 75–87 (2014)

23.76 Z. Chen, T.I. Um, H. Bart-Smith: Bio-inspired robotic manta ray powered by ionic polymer–metal composite artificial muscles, Int. J. Smart Nano Mater. **3**, 296–308 (2012)

23.77 H.-J. Kim, S.-H. Song, S.-H. Ahn: A turtle-like swimming robot using a smart soft composite (SSC) structure, Smart Mater. Struct. **22**, 014007 (2013)

23.78 C.J. Esposito, J.L. Tangorra, B.E. Flammang, G.V. Lauder: A robotic fish caudal fin: Effects of stiffness and motor program on locomotor performance, J. Exp. Biol. **215**, 56–67 (2012)

23.79 H. Prahlad, R. Pelrine, S. Stanford, J. Marlow, R. Kornbluh: Electroadhesive robots—wall climbing robots enabled by a novel, robust, and electrically controllable adhesion technology, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (2008) pp. 3028–3033

23.80 P. Birkmeyer, A.G. Gillies, R.S. Fearing: CLASH: Climbing vertical loose cloth, Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS) (2011) pp. 5087–5093

23.81 K. Streitlien, G.S. Triantafyllou, M.S. Triantafyllou: Efficient foil propulsion through vortex control, AIAA J. **34**, 2315–2319 (1996)

23.82 H. Morikawa, S. Nakao, S.-I. Kobayashi: Experimental study on oscillating wing for propulsor with bending mechanism modeled on caudal muscle-skeletal structure of tuna, Jap. Soc. Mech. Eng. C **44**, 1117–1124 (2001)

23.83 R. Fan, J. Yu, L. Wang, G. Xie, Y. Fang, Y. Hu: Optimized design and implementation of biomimetic robotic dolphin, IEEE Int. Conf. Robotics Biomimet. (ROBIO) (2005) pp. 484–489

23.84 T. Salumäe, M. Kruusmaa: A flexible fin with bio-inspired stiffness profile and geometry, J. Bionic Eng. **8**, 418–428 (2011)

23.85 P.V. y Alvarado, K. Youcef-Toumi: Design of machines with compliant bodies for biomimetic locomotion in liquid environments, J. Dyn. Syst. Meas. Contr. **128**, 3–13 (2006)

23.86 U. Scarfogliero, C. Stefanini, P. Dario: Design and development of the long-jumping, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (2007) pp. 467–472

23.87 M. Kovac, M. Fuchs, A. Guignard, J.-C. Zufferey, D. Floreano: A miniature 7g jumping robot, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (2008) pp. 373–378

23.88 Y.-J. Park, T.M. Huh, D. Park, K.-J. Cho: Design of a variable-stiffness flapping mechanism for maximizing the thrust of a bio-inspired underwater robot, Bioinsp. Biomimet. **9**, 036002 (2014)

23.89 W.-S. Chu, K.-T. Lee, S.-H. Song, M.-W. Han, J.-Y. Lee, H.-S. Kim, M.S. Kim, Y.J. Park, K.J. Cho, S.H. Anh: Review of biomimetic underwater robots using smart actuators, Int. J. Prec. Eng. Manuf. **13**, 1281–1292 (2012)

23.90 Z. Wang, G. Hang, Y. Wang, J. Li, W. Du: Embedded SMA wire actuated biomimetic fin: A module for biomimetic underwater propulsion, Smart Mater. Struc. **17**, 025039 (2008)

23.91 G.V. Lauder, J. Lim, R. Shelton, C. Witt, E. Anderson, J.L. Tangorra: Robotic models for studying undulatory locomotion in fishes, Mar. Technol. Soc. J. **45**, 41–55 (2011)

23.92 F. Li, G. Bonsignori, U. Scarfogliero, D. Chen, C. Stefanini, W. Liu, P. Dario, F. Xin: Jumping mini-robot with bio-inspired legs, IEEE Int. Conf. Robotics Biomimet. (ROBIO) (2009) pp. 933–938

23.93 B.G.A. Lambrecht, A.D. Horchler, R.D. Quinn: A small, insect-inspired robot that runs and jumps, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (2005) pp. 1240–1245

23.94 J. Zhao, J. Xu, B. Gao, N. Xi, F.J. Cintrón, M.W. Mutka, X. Li: MSU Jumper: A single-motor-actuated miniature steerable jumping robot, IEEE Trans. Robotics **29**, 602–614 (2013)

23.95 J. Zhao, W. Yan, N. Xi, M.W. Mutka, L. Xiao: A miniature 25 grams running and jumping robot, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (2014)

23.96 R. Armour, K. Paskins, A. Bowyer, J. Vincent, W. Megill: Jumping robots: A biomimetic solution to locomotion across rough terrain, Bioinsp. Biomimet. **2**, S65–S82 (2007)

Avidbots Ex1016
Page 612 of 2154

23.97 M. Noh, S.-W. Kim, S. An, J.-S. Koh, K.-J. Cho: Flea-inspired catapult mechanism for miniature jumping robots, IEEE Trans. Robotics **28**, 1007–1018 (2012)

23.98 J.-S. Koh, S.-P. Jung, M. Noh, S.-W. Kim, K.-J. Cho: Flea inspired catapult mechanism with active energy storage and release for small scale jumping robot, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (2013) pp. 26–31

23.99 J.-S. Koh, S.-P. Jung, R.J. Wood, K.-J. Cho: A jumping robotic insect based on a torque reversal catapult mechanism, Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS) (2013) pp. 3796–3801

23.100 A. Yamada, M. Watari, H. Mochiyama, H. Fujimoto: An asymmetric robotic catapult based on the closed elastica for jumping robot, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (2008) pp. 232–237

23.101 A.P. Gerratt, S. Bergbreiter: Incorporating compliant elastomers for jumping locomotion in microrobots, Smart Mater. Struct. **22**, 014010 (2013)

23.102 R. Niiyama, A. Nagakubo, Y. Kuniyoshi: Mowgli: A bipedal jumping and landing robot with an artificial musculoskeletal system, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (2007) pp. 2546–2551

23.103 E.W. Hawkes, E.V. Eason, A.T. Asbeck, M.R. Cutkosky: The gecko's toe: Scaling directional adhesives for climbing applications, IEEE/ASME Trans. Mechatron. **18**, 518–526 (2013)

23.104 E.W. Hawkes, D.L. Christensen, E.V. Eason, M.A. Estrada, M. Heverly, E. Hilgemann, J. Hao, M.T. Pope, A. Parness, M.R. Cutkosky: Dynamic surface grasping with directional adhesion, Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS) (2013) pp. 5487–5493

23.105 A.L. Desbiens, A.T. Asbeck, M.R. Cutkosky: Landing, perching and taking off from vertical surfaces, Int. J. Robotics Res. **30**, 355–370 (2011)

23.106 A. Parness, M. Frost, N. Thatte, J.P. King, K. Witkoe, M. Nevarez, M. Garrett, H. Aghazarian, B. Kennedy: Gravity-independent rock-climbing robot and a sample acquisition tool with microspine grippers, J. Field Robotics **30**, 897–915 (2013)

23.107 B.A. Trimmer, A.E. Takesian, B.M. Sweet, C.B. Rogers, D.C. Hake, D.J. Rogers: Caterpillar locomotion: A new model for soft-bodied climbing and burrowing robots, 7th Int. Symp. Technol. Mine Problem (2006) pp. 1–10

23.108 G.-P. Jung, J.-S. Koh, K.-J. Cho: Underactuated adaptive gripper using flexural buckling, IEEE Trans. Robotics **29**(6), 1396 (2013)

23.109 M. Calisti, M. Giorelli, G. Levy, B. Mazzolai, B. Hochner, C. Laschi, P. Dario: An octopus-bioinspired solution to movement and manipulation for soft robots, Bioinsp. Biomimet. **6**, 036002 (2011)

23.110 S.-W. Kim, J.-S. Koh, J.-G. Lee, J. Ryu, M. Cho, K.-J. Cho: Flytrap-inspired robot using structurally integrated actuation based on bistability and a developable surface, Bioinsp. Biomimet. **9**, 036004 (2014)

23.111 C.E. Doyle, J.J. Bird, T.A. Isom, J.C. Kallman, D.F. Bareiss, D.J. Dunlop, R.J. King, J.J. Abbott, M.A. Minor: An avian-inspired passive mechanism for quadrotor perching, IEEE/ASME Trans. Mechatron. **18**, 506–517 (2013)

23.112 M. Kovač, J. Germann, C. Hürzeler, R.Y. Siegwart, D. Floreano: A perching mechanism for micro aerial vehicles, J. Micro-Nano Mechatron. **5**, 77–91 (2009)

23.113 R. Merz, F. Prinz, K. Ramaswami, M. Terk, L. Weiss: Shape deposition manufacturing, Proc. Solid Freeform Fabric. Symp., University of Texas at Austin (1994) pp. 1–8

23.114 S.A. Bailey, J.G. Cham, M.R. Cutkosky, R.J. Full: Biomimetic robotic mechanisms via shape deposition manufacturing, Robotics Res. Int. Symp. (2000) pp. 403–410

23.115 X. Li, A. Golnas, F.B. Prinz: Shape deposition manufacturing of smart metallic structures with embedded sensors, SPIE Proc. 7th Annu. Int. Symp. Smart Struct. Mater. (International Society for Optics and Photonics, Bellingham 2000) pp. 160–171

23.116 K.G. Marra, J.W. Szem, P.N. Kumta, P.A. DiMilla, L.E. Weiss: In vitro analysis of biodegradable polymer blend/hydroxyapatite composites for bone tissue engineering, J. Biomed. Mater. Res. **47**, 324–335 (1999)

23.117 J.G. Cham, S.A. Bailey, J.E. Clark, R.J. Full, M.R. Cutkosky: Fast and robust: Hexapedal robots via shape deposition manufacturing, Int. J. Robotics Res. **21**, 869–882 (2002)

23.118 A.M. Dollar, R.D. Howe: A robust compliant grasper via shape deposition manufacturing, IEEE/ASME Trans. Mechatron. **11**, 154–161 (2006)

23.119 M. Binnard, M.R. Cutkosky: Design by composition for layered manufacturing, J. Mech. Des. **122**, 91–101 (2000)

23.120 Y.-L. Park, K. Chau, R.J. Black, M.R. Cutkosky: Force sensing robot fingers using embedded fiber Bragg grating sensors and shape deposition manufacturing, IEEE Int. Conf. Robotics Autom. (ICRA) (2007) pp. 1510–1516

23.121 D. Shin, I. Sardellitti, Y.-L. Park, O. Khatib, M. Cutkosky: Design and control of a bio-inspired human-friendly robot, Int. J. Robotics Res. **29**, 571–584 (2010)

23.122 R.S. Fearing, K.H. Chiang, M.H. Dickinson, D.L. Pick, M. Sitti, J. Yan: Wing transmission for a micromechanical flying insect, Proc. IEEE Int. Conf. Robotics Autom. (ICRA), Vol. 2 (2000) pp. 1509–1516

23.123 J. Yan, R.J. Wood, S. Avadhanula, M. Sitti, R.S. Fearing: Towards flapping wing control for a micromechanical flying insect, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (2001) pp. 3901–3908

23.124 R.J. Wood, S. Avadhanula, M. Menon, R.S. Fearing: Microrobotics using composite materials: The micromechanical flying insect thorax, Proc. IEEE Int. Conf. Robotics Autom. (ICRA), Vol. 2 (2003) pp. 1842–1849

Avidbots Ex1016
Page 613 of 2154

23.125 K.Y. Ma, P. Chirarattananon, S.B. Fuller, R.J. Wood: Controlled flight of a biologically inspired, insect-scale robot, Science **340**, 603–607 (2013)

23.126 J. Whitney, P. Sreetharan, K. Ma, R. Wood: Pop-up book MEMS, J. Micromech. Microeng. **21**, 115021 (2011)

23.127 P.S. Sreetharan, J.P. Whitney, M.D. Strauss, R.J. Wood: Monolithic fabrication of millimeter-scale machines, J. Micromech. Microeng. **22**, 055027 (2012)

23.128 C. Majidi: Soft robotics: A perspective – Current trends and prospects for the future, Soft Robotics **1**, 5–11 (2013)

23.129 Y. Xia, G.M. Whitesides: Soft lithography, Annu. Rev. Mater. Sci. **28**, 153–184 (1998)

23.130 F. Ilievski, A.D. Mazzeo, R.F. Shepherd, X. Chen, G.M. Whitesides: Soft robotics for chemists, Angew. Chem. **123**, 1930–1935 (2011)

23.131 R.F. Shepherd, F. Ilievski, W. Choi, S.A. Morin, A.A. Stokes, A.D. Mazzeo, X. Chen, M. Wang, G.M. Whitesides: Multigait soft robot, Proc. Natl. Acad. Sci. **108**, 20400–20403 (2011)

23.132 B.C.-M. Chang, J. Berring, M. Venkataram, C. Menon, M. Parameswaran: Bending fluidic actuator for smart structures, Smart Mater. Struct. **20**, 035012 (2011)

23.133 B. Chang, A. Chew, N. Naghshineh, C. Menon: A spatial bending fluidic actuator: Fabrication and quasi-static characteristics, Smart Mater. Struct. **21**, 045008 (2012)

23.134 B. Finio, R. Shepherd, H. Lipson: Air-Powered Soft Robots for K-12 Classrooms, IEEE Proc. Integr. STEM Edu. Conf. (ISEC) (2013) pp. 1–6

23.135 S.A. Morin, R.F. Shepherd, S.W. Kwok, A.A. Stokes, A. Nemiroski, G.M. Whitesides: Camouflage and display for soft machines, Science **337**, 828–832 (2012)

23.136 R.V. Martinez, J.L. Branch, C.R. Fish, L. Jin, R.F. Shepherd, R. Nunes, Z. Suo, G.M. Whitesides: Robotic tentacles with three-dimensional mobility based on flexible elastomers, Adv. Mater. **25**, 205–212 (2013)

23.137 R.V. Martinez, C.R. Fish, X. Chen, G.M. Whitesides: Elastomeric origami: Programmable paper-elastomer composites as pneumatic actuators, Adv. Funct. Mater. **22**, 1376–1384 (2012)

23.138 S.W. Kwok, S.A. Morin, B. Mosadegh, J.H. So, R.F. Shepherd, R.V. Martinez, B. Smith, F.C. Simeone, A.A. Stokes, G.M. Whitesides: Magnetic assembly of soft robots with hard components, Adv. Funct. Mater. **24**, 2180–2187 (2013)

23.139 B. Mosadegh, P. Polygerinos, C. Keplinger, S. Wennstedt, R.F. Shepherd, U. Gupta, J. Shim, K. Bertoldi, C.J. Walsh, G.M. Whitesides: Pneumatic networks for soft robotics that actuate rapidly, Adv. Funct. Mater. **24**, 2163–2170 (2013)

23.140 M. Cianchetti, A. Arienti, M. Follador, B. Mazzolai, P. Dario, C. Laschi: Design concept and validation of a robotic arm inspired by the octopus, Mater. Sci. Eng. C **31**, 1230–1239 (2011)

23.141 T. Umedachi, V. Vikas, B.A. Trimmer: Highly deformable 3-D printed soft robot generating inching and crawling locomotions with variable friction legs, Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS) (2013) pp. 4590–4595

Avidbots Ex1016
Page 614 of 2154

Multimedia Contents

24. Wheeled Robots

Woojin Chung, Karl Iagnemma

The purpose of this chapter is to introduce, analyze, and compare various wheeled mobile robots (WMRs) and to present several realizations and commonly encountered designs. The mobility of WMR is discussed on the basis of the kinematic constraints resulting from the pure rolling conditions at the contact points between the wheels and the ground. Practical robot structures are classified according to the number of wheels, and features are introduced focusing on commonly adopted designs. Omnimobile robot and articulated robots realizations are described. Wheel–terrain interaction models are presented in order to compute forces at the contact interface. Four possible wheel–terrain interaction cases are shown on the basis of relative stiffness of the wheel and terrain. A suspension system is required to move on uneven surfaces. Structures, dynamics, and important features of commonly used suspensions are explained.

24.1 **Overview** 575
24.2 **Mobility of Wheeled Robots** 576
24.2.1 Types of Wheels 576
24.2.2 Kinematic Constraints 578
24.2.3 Robot Configuration Variables 580
24.2.4 Restriction on Robot Mobility 580
24.2.5 Characterization of Robot Mobility 581
24.2.6 Five Classes of Wheeled Mobile Robots 581
24.3 **Wheeled Robot Structures** 582
24.3.1 Robots with One Wheel 582
24.3.2 Robots with Two Wheels 582
24.3.3 Robots with Three Wheels 583
24.3.4 Robots with Four Wheels 585
24.3.5 Special Applications of Wheeled Robots 585
24.4 **Wheel–Terrain Interaction Models** 586
24.4.1 Rigid Wheels on Rigid Terrain 586
24.4.2 Rigid Wheels on Deformable Terrain 586
24.4.3 Deformable Tires on Deformable Terrain 587
24.4.4 Deformable Tires on Rigid Terrain 589
24.5 **Wheeled Robot Suspensions** 589
24.5.1 Passive Suspension Dynamics 590
24.5.2 Semiactive Suspension Dynamics 591
24.6 **Conclusions** 592
Video-References 592
References 593

24.1 Overview

Wheeled robots have been widely used to achieve mobility because there are many advantages including the simple structure, energy efficiency, fast speed, low fabrication cost, and so forth. The purpose of this chapter is to provide a general description of wheeled mobile robots, to discuss their properties from the mobility point of view, to describe the most commonly encountered realizations of such robots, and to explain wheel–terrain interaction models and suspension systems.

The chapter is organized as follows: Sect. 24.2 is devoted to the characterization of the restriction of robot motion induced by these pure rolling conditions. We first describe different types of wheels used in the construction of mobile robots and derive the corresponding kinematic constraints. This allows us to characterize the mobility of a robot equipped with several wheels of these different types, and we show that these robots can be classified only into five categories, corresponding to two mobility indices.

Avidbots Ex1016
Page 615 of 2154

In Sect. 24.3, we present several realizations of wheeled mobile robots by the use of various types of wheels. Structures and features of widely used wheel mechanisms are explained. Design issues of wheeled robots include capability of omnidirectional movement, fabrication cost, difficulty of control, and terrain traversability.

Section 24.4 explains wheel–terrain interaction models. It is shown that four possible wheel–terrain interaction cases can be obtained according to the relative stiffness of the wheel and terrain. The presented analytical models are broadly suitable for analysis, simulation, design, and control purposes.

Section 24.5 presents suspensions for wheeled robots. A suspension is a system of linkages, springs, dampers, and actuators that governs the relative motion between a robot's wheels and body. Suspension mechanisms are useful when driving over uneven surfaces. Common methods for modeling passive and semiactive suspensions are presented. The analysis of these models allows for principled selection of springs, dampers, and active or semiactive actuators that form the basis of nearly all wheeled robot suspensions.

Finally, some concluding remarks are given in Sect. 24.6.

24.2 Mobility of Wheeled Robots

In this section, we describe a variety of wheels and wheel implementations in mobile robots. We discuss the restriction of robot mobility implied by the use of these wheels and deduce a classification of robot mobility allowing one to characterize robot mobility fully, whatever the number and type of the wheels.

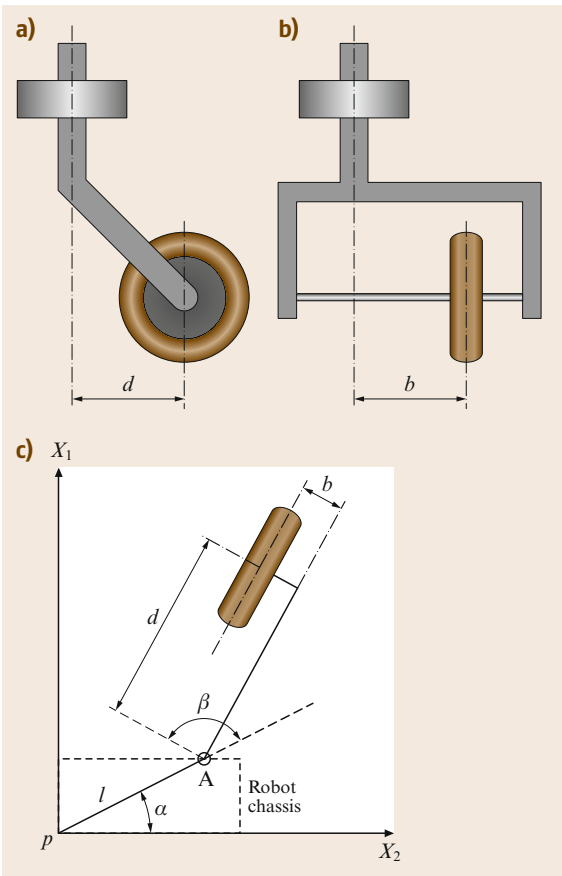


Fig.24.1a–c The general design of a standard wheel. **(a)** Side view, **(b)** front view, and **(c)** top view

24.2.1 Types of Wheels

In order to achieve robot locomotion, wheeled mobile robots are widely used in many applications. In general, wheeled robots consume less energy and move faster than other locomotion mechanisms (e.g., legged robots or tracked vehicles). From the viewpoint of control, less control effort is required, owing to their simple mechanisms and reduced stability problems. Although it is difficult to overcome rough terrain or uneven ground conditions, wheeled mobile robots are suitable for a large class of target environments in practical applications. When we think of a single-wheel design, there are two candidates: a standard wheel or a special wheel. A standard wheel can be understood as a conventional tire. Special wheels possess unique mechanical structures including rollers or spheres. Figure 24.1 shows the general design of a standard wheel. Three conditions should be defined for a standard wheel design:

1. Determination of the two offsets d and b
2. Mechanical design that allows steering motion or not (i. e., to fix the wheel orientation or not)
3. Determination of steering and driving actuation (i. e., active or passive drive).

Condition 1 is the kinematic parameter design problem for a single standard wheel. The parameter d can be either 0 or some positive constant. Parameter b is the lateral offset of the wheel and is usually set to zero. In a special design, a nonzero b may be selected to obtain pure rolling contact between the wheel and ground without causing rotational slip at the contact point. However, this is rarely used and we mainly consider the case of zero lateral offset b.

Condition 2 is a design problem for whether the wheel orientation can be changed or not. If the steering axis is fixed, the wheel provides a velocity constraint on the driving direction. Condition 3 is the design problem of whether to actuate steering or driving motion by actuators or to drive steering or motion passively.

If steering motion is allowed, the offset d plays a significant role in the kinematic modeling. For a conventional caster wheel (i. e., an off-centered orientable wheel), there is a nonzero offset d. Point A in Fig. 24.1 indicates the location of the joint connecting the wheel module to the robot chassis. Two orthogonal linear velocity components at point A are obtained by a caster wheel, which results from the steering and driving motions of the wheel module. This implies that a passive caster wheel does not provide an additional velocity constraint on the robot's motion. If a caster wheel is equipped with two actuators that drive steering and driving motions independently, holonomic omnidirectional movement can be achieved because any desired velocity at point A can be generated by solving the inverse kinematics problem.

If the offset d is set to zero, the allowable velocity direction at point A is limited to the wheel orientation. In such a case, the steering motion should not be passive because the wheel orientation cannot be changed passively. However, the driving velocity can be determined passively by the actuation of other wheels. Wheel orientation should be actively steered to the desired velocity direction due to the nonholonomic velocity constraint. This implies that the wheel orientation should be aligned before movement.

In summary, four types of standard wheels are commonly used. First is a passively driven wheel with a fixed steering axis. Second is a passive caster wheel with offset d. Third is an active caster wheel with offset d, where the steering and driving motions are controlled by actuators. An example of caster wheels is shown in VIDEO 325. The fourth is an active orientable wheel with zero offset d, where steering and driving motions are driven by actuators. The structures of each wheel type are shown in Fig. 24.2. The kinematics and constraints of those wheels will be explained in detail in Sect. 24.2.2.

Although standard wheels are advantageous because of their simple structure and good reliability, the nonholonomic velocity constraint (i. e., no sideslip condition) limits robot motion. On the other hand, special wheels can be employed in order to obtain omnidirectional motion of a mobile robot (omnimobile robot), i. e., to ensure three degrees of freedom for plane motion. We consider two typical designs of special wheels: the Swedish wheel and the spherical wheel. Figure 24.3a and VIDEO 328 show the

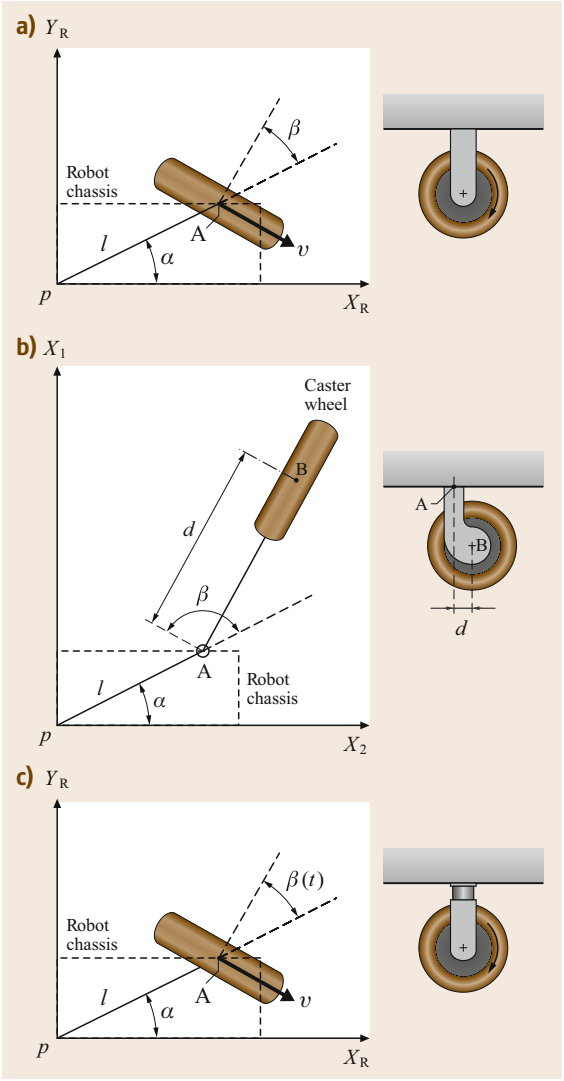


Fig.24.2a–c Structures of standard wheels. **(a)** Passive fixed wheel, **(b)** passive or active, off-centered orientable wheel, and **(c)** active orientable wheel without offsets

Swedish wheel. Small passive free rollers are located along the outer rim of the wheel. Free rollers are employed in order to eliminate the nonholonomic velocity constraint. Passive rollers are free to rotate around the axis of rotation, which results in lateral motion of the wheel. As a result, a driving velocity should be controlled, while the lateral velocity is passively determined by the actuation of the other wheels. A similar design can be seen in VIDEO 327.

A spherical wheel is shown in Fig. 24.3c. The rotation of the sphere is constrained by rollers that make rolling contact with the sphere. The rollers can be divided into driving and supporting rollers. The sphere is

Avidbots Ex1016
Page 617 of 2154

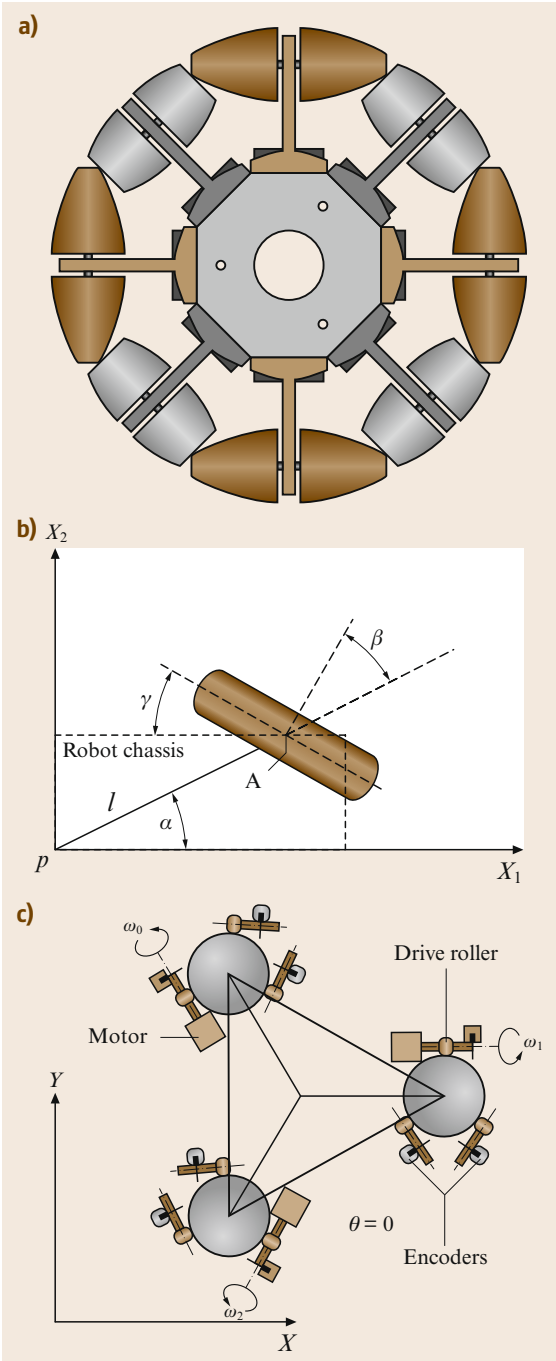


Fig. 24.3 (a) Swedish wheel, (b) attachment of a Swedish wheel, and (c) spherical wheel (after [24.1])

driven by the actuation of the driving rollers, whereas the rolling contacts provide nonholonomic constraints, and the resultant motion of the sphere module becomes holonomic. This implies that the robot can be moved with any desired linear/angular velocities at any time. By using the spherical wheel, a holonomic omnidirectional mobile robot can be developed and the robot achieves smooth and continuous contact between the sphere and the ground. However, the design of the sphere-supporting mechanism is difficult and the payload must be quite low due to the point contact. Another drawback is that the surface of the sphere can be polluted when traveling over dirty ground and it is difficult to overcome irregular ground conditions. These drawbacks limit the practical application of the spherical wheel. An example of the use of spherical wheels can be found in [24.1] and [24.2]. The spherical structure can also be applied to special robotic transmissions; examples include the nonholonomic manipulator in [24.3] and the passive haptic system in [24.4].

24.2.2 Kinematic Constraints

We assume, as a first step, that the mobile robot under study is made up of a rigid cart equipped with nondeformable wheels, and that it is moving on a horizontal plane. The position of the robot on the plane is described, with respect to an arbitrary inertial frame, by the *posture* vector $\boldsymbol{\xi} = (x \quad y \quad \theta)^{\mathrm{T}}$, where x and y are the coordinates of a reference point P of the robot cart, while θ describes the orientation of a mobile frame attached to the robot, with respect to the inertial frame (Fig. 24.4).

We assume that, during motion, the plane of each wheel remains vertical and the wheel rotates around its horizontal axle, whose orientation with respect to the cart can be fixed or varying. We distinguish between the two basic classes of idealized wheels, namely conventional and the Swedish wheels. In each case,

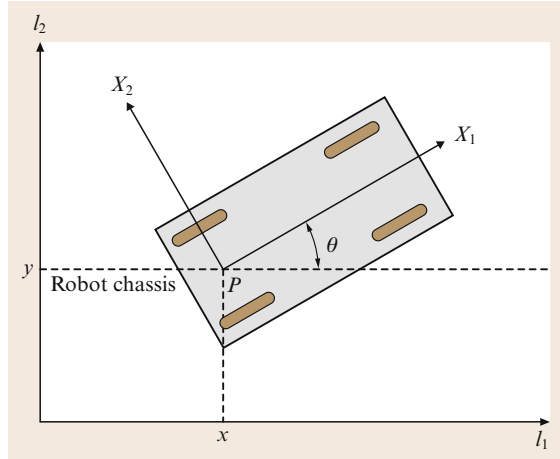


Fig. 24.4 The posture definition of a mobile robot on a plane

it is assumed that the contact between the wheel and the ground is reduced to a single point. The kinematic constraints result from the fact that the velocity of the material point of the wheel in contact with the ground is equal to zero.

For a conventional wheel, the kinematic constraints imply that the velocity of the center of the wheel is parallel to the wheel plane (*nonslip condition*) and is proportional to the wheel rotation velocity (*pure rolling condition*). For each wheel, the kinematic constraints therefore result in two independent conditions. For a Swedish wheel, due to the relative rotation of the rollers with respect to the wheel, only one of the velocity components of the wheel contact point is zero. The direction of this zero component is fixed with respect to the wheel plane and depends on the wheel construction. For such wheels, the kinematic constraints result in only one condition.

Conventional Wheels

We now derive the general form of the kinematic constraints for a conventional wheel.

As shown in Fig. 24.2, there are several variations of the conventional wheel design. First, we focus on the off-centered orientable wheel in Fig. 24.2b. The center of the wheel, B, is connected to the cart by a rigid rod from A (a fixed point on the cart) to B, aligned with the wheel plane. The rod, whose length is denoted by d, can rotate around a fixed vertical axle at point A. The position of A is specified by two constant polar coordinates, l and α, with respect to the reference point P. The rotation of the rod with respect to the cart is represented by the angle β. The radius of the wheel is denoted by r, and its angle of rotation around its horizontal axle is denoted by φ. The description therefore involves four constant parameters: α, l, r, and d, and two variables: $\varphi(t)$ and $\beta(t)$.

With these notations the kinematic constraints are derived as follows.

We make the derivation explicit for the general situation corresponding to a caster wheel (Fig. 24.2b). For fixed or steering wheels one just has to consider either the case $d = 0$ and constant β (fixed wheels), or $d = 0$ and variable β (steering wheels).

First, we evaluate the velocity of the center of the wheel, which results from the following vector expression $\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{OB} = \frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{OP} + \frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{PA} + \frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{AB}$. The two components of this vector in the robot frame are expressed as

$$\dot{x}\cos\theta + \dot{y}\sin\theta - l\dot{\theta}\sin\alpha + (\dot{\theta}+\dot{\beta})\mathrm{d}\cos(\alpha+\beta)$$

and

$$-\dot{x}\sin\theta + \dot{y}\cos\theta - l\,\dot{\theta}\cos\alpha + (\dot{\theta}+\dot{\beta})\mathrm{d}\sin(\alpha+\beta)\,.$$

The projections of this vector onto the direction of the wheel plane, i. e., onto the vector $(\cos(\alpha+\beta-\pi/2), \sin(\alpha+\beta-\pi/2))$ and the vector of the wheel axle $(\cos(\alpha+\beta), \sin(\alpha+\beta))$, are $r\dot{\boldsymbol{\varphi}}$ and 0, respectively, corresponding to the pure rolling and nonslip conditions.

After some manipulations, these conditions can be rewritten in the following compact form.

Pure Rolling Condition.

$$(-\sin(\alpha+\beta)\ \ \cos(\alpha+\beta)\ \ l\cos\beta)\mathbf{R}(\theta)\dot{\boldsymbol{\xi}} + r\dot{\boldsymbol{\varphi}} = 0\,. \tag{24.1}$$

Nonslip Condition.

$$(-\cos(\alpha+\beta)\ \ \sin(\alpha+\beta)\mathrm{d} + l\sin\beta)\mathbf{R}(\theta)\dot{\boldsymbol{\xi}} + d\dot{\boldsymbol{\beta}} = 0\,. \tag{24.2}$$

In the earlier given expressions, $\mathbf{R}(\theta)$ is the orthogonal rotation matrix expressing the orientation of the robot with respect to the inertial frame, i. e.,

$$\mathbf{R}(\theta) = \begin{pmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}. \tag{24.3}$$

As said before, these general expressions can be simplified for different types of conventional wheels.

For *fixed wheels*, the center of the wheel is fixed with respect to the cart and the wheel orientation is constant. This corresponds to a constant value of β and $d = 0$ (Fig. 24.2a). The nonslip equation (24.2) then reduces to

$$(\cos(\alpha+\beta)\ \ \sin(\alpha+\beta)\ \ l\sin\beta)\mathbf{R}(\theta)\dot{\boldsymbol{\xi}} = 0\,. \tag{24.4}$$

For *steering wheels*, the center of the wheel is also fixed with respect to the cart (i. e., $d = 0$), with β time-varying, so the nonslip equation takes the form (24.2). This structure was already introduced in Fig. 24.2c.

The situation described by (24.1) and (24.2), with a nonzero-length rod AB and time-varying orientation angle β corresponds to *caster wheels*.

Swedish Wheels. The position of a Swedish wheel with respect to the cart is described, as for fixed wheels, by three constant parameters: α, β, and l. An additional parameter is required to characterize the direction, with respect to the wheel plane, of the zero component of the velocity at the contact point of the wheel. This parameter is γ, which is the angle between the axle of the rollers and the wheel plane (Fig. 24.3b).

The kinematic constraints now impose only one condition

$$[-\sin(\alpha+\beta+\gamma) \quad \cos(\alpha+\beta+\gamma) \quad l\cos(\beta+\gamma)] \times \mathbf{R}(\theta)\dot{\boldsymbol{\xi}} + r\cos\gamma\,\dot{\boldsymbol{\varphi}} = 0 \tag{24.5}$$

24.2.3 Robot Configuration Variables

We now consider a wheeled robot equipped with N wheels of the earlier described types. We use the following subscripts to identify quantities related to these four types: f for fixed wheels, s for steering wheels, c for caster wheels, and sw for Swedish wheels. The number of wheels of each type are denoted by N_f, N_s, N_c, and N_sw, with $N = N_\mathrm{f} + N_\mathrm{s} + N_\mathrm{c} + N_\mathrm{sw}$.

The configuration of the robot is fully described by the following generalized coordinate vector:

- *Posture coordinates*: the posture vector $\boldsymbol{\xi}(t) = (x(t)\ \ y(t)\ \ \theta(t))^\mathrm{T}$
- *Orientation coordinates*: the $N_\mathrm{s}+N_\mathrm{c}$ orientation angles of the steering and caster wheels, i. e., $\boldsymbol{\beta}(t) = (\beta_\mathrm{s}(t)\ \beta_\mathrm{c}(t))^\mathrm{T}$
- *Rotation coordinates*: the N rotation angles of the wheels, i. e., $\boldsymbol{\varphi}(t) = (\varphi_\mathrm{f}(t)\ \varphi_\mathrm{s}(t)\ \varphi_\mathrm{c}(t)\ \varphi_\mathrm{sw}(t))^\mathrm{T}$.

This whole set of coordinates is termed as the set of configuration coordinates. The total number of configuration coordinates is $N_\mathrm{f} + 2N_\mathrm{s} + 2N_\mathrm{c} + N_\mathrm{sw} + 3$.

24.2.4 Restriction on Robot Mobility

The *pure rolling conditions* for fixed, steering, and caster wheels, as well as the constraints relative to the Swedish wheels, can be written in the following compact form

$$\mathbf{J}_1(\beta_\mathrm{s}, \beta_\mathrm{c})\mathbf{R}(\theta)\dot{\boldsymbol{\xi}} + \mathbf{J}_2\dot{\boldsymbol{\varphi}} = 0\,, \tag{24.6}$$

with

$$\mathbf{J}_1(\boldsymbol{\beta}_\mathrm{s}, \boldsymbol{\beta}_\mathrm{c}) = \begin{pmatrix} \mathbf{J}_{1\mathrm{f}} \\ \mathbf{J}_{1\mathrm{s}}(\beta_\mathrm{s}) \\ \mathbf{J}_{1\mathrm{c}}(\beta_\mathrm{c}) \\ \mathbf{J}_{1\mathrm{sw}} \end{pmatrix}.$$

In this expression, $\mathbf{J}_{1\mathrm{f}}$, $\mathbf{J}_{1\mathrm{s}}(\beta_\mathrm{s})$, $\mathbf{J}_{1\mathrm{c}}(\beta_\mathrm{c})$, and $\mathbf{J}_{1\mathrm{sw}}$ are, respectively, $(N_\mathrm{f}\times 3)$, $(N_\mathrm{s}\times 3)$, $(N_\mathrm{c}\times 3)$, and $(N_\mathrm{sw}\times 3)$ matrices, whose forms derive directly from the kinematic constraints, while $\mathbf{J}_2$ is a constant $(N\times N)$ diagonal matrix whose entries are the radii of the wheels, except for the radii of the Swedish wheels which are multiplied by $\cos\gamma$.

The value $\gamma = \pi/2$ would correspond to the direction of the zero component of the velocity being orthogonal to the plane of the Swedish wheel. Such a wheel would be subject to a constraint identical to the nonslip condition for a conventional wheel; hence, losing the benefit of implementing a Swedish wheel. This implies that $\gamma \neq \frac{\pi}{2}$ and that $\mathbf{J}_2$ is a nonsingular matrix.

The *nonslip conditions* for caster wheels can be summarize as

$$\mathbf{C}_{1\mathrm{c}}(\beta_\mathrm{c})\mathbf{R}(\theta)\dot{\boldsymbol{\xi}} + \mathbf{C}_{2\mathrm{c}}\dot{\beta}_\mathrm{c} = 0\,, \tag{24.7}$$

where $\mathbf{C}_{1\mathrm{c}}(\beta_\mathrm{c})$ is a $(N_\mathrm{c}\times 3)$ matrix, whose entries derive from the nonslip constraints (24.2), while $\mathbf{C}_{2\mathrm{c}}$ is a constant diagonal nonsingular matrix, whose entries are equal to d.

The last constraints relate to the *nonslip conditions for fixed and steering wheels*. They can be summarized as

$$\mathbf{C}_1^*(\beta_\mathrm{s})\mathbf{R}(\theta)\dot{\boldsymbol{\xi}} = 0\,, \tag{24.8}$$

where

$$\mathbf{C}_1^*(\beta_\mathrm{s}) = \begin{pmatrix} \mathbf{C}_{1\mathrm{f}} \\ \mathbf{C}_{1\mathrm{s}}(\beta_\mathrm{s}) \end{pmatrix},$$

where $\mathbf{C}_{1\mathrm{f}}$ and $\mathbf{C}_{1\mathrm{s}}(\beta_\mathrm{s})$ are, respectively, $(N_\mathrm{f}\times 3)$ and $(N_\mathrm{s}\times 3)$ matrices.

It is important to point out that the restrictions on robot mobility result only from the conditions (24.8) involving the fixed and the steering wheels. These conditions imply that the vector $\mathbf{R}(\theta)\dot{\boldsymbol{\xi}}$ belongs to $N[\mathbf{C}_1^*(\beta_\mathrm{s})]$, the null space of the matrix $\mathbf{C}_1^*(\beta_\mathrm{s})$. For any $\mathbf{R}(\theta)\dot{\boldsymbol{\xi}}$ satisfying this condition, there exists a vector $\dot{\boldsymbol{\varphi}}$ and a vector $\dot{\boldsymbol{\beta}}_\mathrm{c}$ satisfying, respectively, conditions (24.6) and (24.7), because $\mathbf{J}_2$ and $\mathbf{C}_{2\mathrm{c}}$ are nonsingular matrices.

Obviously $\mathrm{rank}[\mathbf{C}_1^*(\beta_\mathrm{s})] \leq 3$. If it is equal to 3 then $\mathbf{R}(\theta)\dot{\boldsymbol{\xi}} = 0$, which means that any motion in the plane is impossible. More generally, restrictions on robot mobility are related to the rank of $\mathbf{C}_1^*(\beta_\mathrm{s})$, as will be discussed as follows in detail.

It is worth noticing that condition (24.8) has a direct geometrical interpretation. At each time instant the motion of the robot can be viewed as an instantaneous rotation about the instantaneous center of rotation (ICR), whose position with respect to the cart can be time varying. At each instant the velocity of any point of the cart is orthogonal to the straight line joining this point and the ICR. This is true, in particular, for the centers of the fixed and steering wheels, which are fixed points of the cart. On the other hand, the nonslip condition implies that the velocity of the wheel center is aligned with the wheel plane. These two facts imply that

the horizontal rotation axles of the fixed and steering wheels intersect at the ICR (Fig. 24.5). This is equivalent to the condition that $\text{rank}[\mathbf{C}_1^*(\beta_s)] \leq 3$.

24.2.5 Characterization of Robot Mobility

As said earlier, the mobility of the robot is directly related to the rank of $\mathbf{C}_1^*(\boldsymbol{\beta}_s)$, which depends on the design of the robot. We define the *degree of mobility* δ_m as

$$\delta_m = 3 - \text{rank}[\mathbf{C}_1^*(\boldsymbol{\beta}_s)] \,. \tag{24.9}$$

Let us first examine the case $\text{rank}(\mathbf{C}_{1f}) = 2$, which implies that the robot has at least two fixed wheels. If there are more than two fixed wheels, their axles intersect at the ICR, whose position with respect to the cart is then fixed in such a way that the only possible motion is a rotation of the cart about this fixed ICR. Obviously, from the user's point of view, such a design is not acceptable. We therefore assume that $\text{rank}(\mathbf{C}_{1f}) \leq 1$.

Moreover, we assume that

$$\text{rank}[\mathbf{C}_1^*(\boldsymbol{\beta}_s)] = \text{rank}(\mathbf{C}_{1f}) + \text{rank}[\mathbf{C}_{1s}(\boldsymbol{\beta}_s)] \leq 2 \,.$$

These two assumptions are equivalent to the following set of conditions:

1. If the robot has more than one fixed wheel, they are all on a single common axle.
2. The centers of the steering wheels do not belong to this common axle of the fixed wheels.
3. The number $\text{rank}[\mathbf{C}_{1s}(\boldsymbol{\beta}_s)]$ is equal to the number of steering wheels that can be oriented independently in order to steer the robot.

We call this number the *degree of steerability*

$$\delta_s = \text{rank}[\mathbf{C}_{1s}(\boldsymbol{\beta}_s)] \,. \tag{24.10}$$

If a robot is equipped with more than δ_s steering wheels, the motion of the extra wheels must be coordinated in order to guarantee the existence of the ICR at each instant.

We conclude that, for wheeled mobile robot of practical interest, the two defined indices, δ_m and δ_s, satisfy the following conditions:

1. The degree of mobility satisfies $1 \leq \delta_m \leq 3$. The upper bound is obvious, while the lower bound means that we consider only cases where motion is possible.
2. The degree of steerability satisfies $0 \leq \delta_s \leq 2$. The upper bound can be reached only for robots without fixed wheels, while the lower bound corresponds to robots without steering wheels.
3. The following is satisfied: $2 \leq \delta_m + \delta_s \leq 3$.

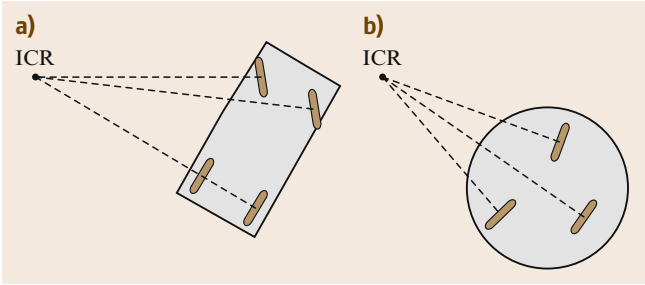


Fig.24.5a,b The instantaneous center of rotation. **(a)** A car-like robot; **(b)** a three-steering-wheels robot

The case $\delta_m + \delta_s = 1$ is not acceptable because it corresponds to the rotation of the robot about a fixed ICR. The cases $\delta_m \geq 2$ and $\delta_s = 2$ are excluded because according to the assumptions, $\delta_s = 2$ implies $\delta_m = 1$. These conditions imply that only five structures are of practical interest, corresponding to the five pairs (δ_m, δ_s) satisfying the aforementioned inequalities, according to the following array

δ_m	3	2	2	1	1
δ_s	0	0	1	1	2

In the following, each type of structure will be designated by using a denomination of the form type (δ_m, δ_s) robot.

24.2.6 Five Classes of Wheeled Mobile Robots

We now briefly describe the five classes of wheeled robot structures, pointing out the mobility restriction inherent to each class. Details and examples can be found in Sect. 24.3 and in [24.5].

Type (3,0) Robots

These robots have no fixed and no steering wheels and are equipped only with Swedish or caster wheels. Such robots are called *omnimobile*, because they have full mobility in the plane, which means that they are able to move in any direction without any reorientation.

Type (2,0) Robots

These robots have no steering wheels, but either one or several fixed wheels with a common axle. Mobility is restricted in the sense that, at a given posture $\xi(t)$, the velocity $\dot{\boldsymbol{\xi}}(t)$ is constrained to belong to a two-dimensional distribution spanned by the vector fields $\mathbf{R}^T(\theta)\boldsymbol{s}_1$ and $\mathbf{R}^T(\theta)\boldsymbol{s}_2$, where $\boldsymbol{s}_1$ and $\boldsymbol{s}_2$ are two constant vectors spanning $N(\mathbf{C}_{1f})$. A typical example of such a robot is the wheelchair.

Avidbots Ex1016
Page 621 of 2154

Type (2,1) Robots

These robots have no fixed wheels and at least one steering wheel. If there is more than one steering wheel, their orientations must be coordinated in such a way that $\text{rank}[\mathbf{C}_{1s}(\boldsymbol{\beta}_s)] = \delta_s = 1$. The velocity $\dot{\boldsymbol{\xi}}(t)$ is constrained to belong to a two-dimensional distribution spanned by the vector fields $\mathbf{R}^T(\theta)\boldsymbol{s}_1(\boldsymbol{\beta}_s)$ and $\mathbf{R}^T(\theta)\boldsymbol{s}_2(\boldsymbol{\beta}_s)$, where $\boldsymbol{s}_1(\beta_s)$ and $\boldsymbol{s}_2(\beta_s)$ are two vectors spanning $N(\mathbf{C}_{1s}(\beta_s))$.

Type (1,1) Robots

These robots have one or several fixed wheels on a single common axle, and also one or several steering wheels, with the conditions that their centers are not located on the common axle of the fixed wheels, and that their orientations are coordinated. The velocity $\dot{\boldsymbol{\xi}}(t)$ is constrained to belong to a one-dimensional distribution parameterized by the orientation angle of one arbitrarily chosen steering wheel. Mobile robots built on the model of a conventional car (often called *car-like* robots) belong to this class.

Type (1,2) Robots

These robots have no fixed wheels, but at least two steering wheels. If there are more than two steering wheels, then their orientation must be coordinated in order to satisfy the condition $\text{rank}[\mathbf{C}_{1s}(\boldsymbol{\beta}_s)] = \delta_s = 2$. The velocity $\dot{\boldsymbol{\xi}}(t)$ is constrained to belong to a one-dimensional distribution parameterized by the orientation angles of two arbitrarily chosen steering wheels.

24.3 Wheeled Robot Structures

There are many design alternatives for wheeled mobile robots. Design problems of a single-body mobile robot include the selection of wheel types, the placement of wheels, and the determination of the kinematic parameters. Design objectives should be specified according to the target environments and tasks, as well as the initial and operational costs of a robot. In this section, robot structures are classified according to the number of wheels, and then features will be introduced focusing on commonly adopted designs.

24.3.1 Robots with One Wheel

A robot with a single wheel is basically unstable without dynamic control in order to maintain its balance of the body. A typical example is a unicycle. As a variation of a unicycle, a robot with a rugby-ball-shaped wheel can be used in order to improve stability in the lateral direction, as studied in [24.6].

A spherical robot can also be considered as a single-wheel robot. A balancing mechanism such as a spinning wheel is employed to achieve dynamic stability. This approach has advantages including high maneuverability and low rolling resistance. However, single-wheel robots are rarely used in practical applications, because additional balancing mechanisms are required, control is difficult, and pose estimation by pure dead reckoning is not available. An example of a spherical robot can be found in [24.7].

24.3.2 Robots with Two Wheels

In general, there are two types of two-wheel robots, as shown in Fig. 24.6. Figure 24.6a shows a bicycle-type robot. It is common to steer a front wheel and to drive a rear wheel. Since the dynamic stability of a bicycle-type robot increases with its speed, a balancing mechanism is not necessarily required. The advantage of this approach is that the robot width can be reduced. However, a bicycle type is rarely used because it cannot maintain its pose when the robot stands still. Figure 24.6b shows an inverted-pendulum-type robot. It is a two-wheel differential drive robot.

It is possible to achieve static stability by accurately placing the center of gravity on the wheel axle. However, it is common to apply dynamic balancing control, which is similar to the conventional control problem for an inverted pendulum. The size of a robot can be re-

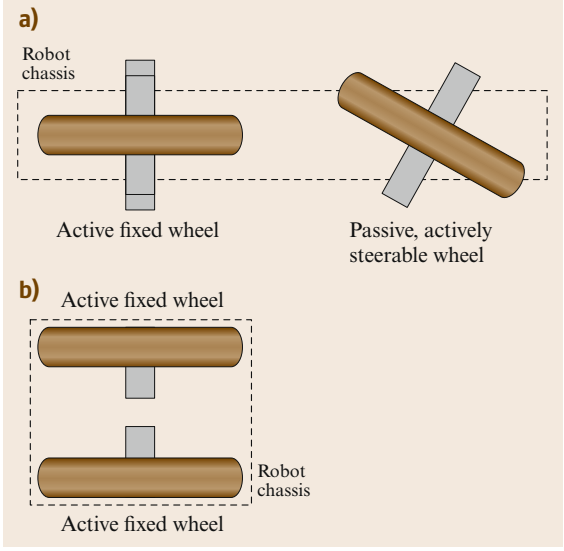


Fig. 24.6 (a) Bicycle-type robot and (b) inverted-pendulum-type robot

duced by using two-wheel robots, when compared with robots with more than three wheels. A typical application of a pendulum-type robot is to design a structure as a four-wheel robot, consisting of two pendulum robots connected. Then, the robot can climb stairs by lifting its front wheels while the robot reaches the stair. A major disadvantage is that control effort is always required for dynamic balancing. Examples of inverted-pendulum-type robots can be found in [24.8] and [24.9].

24.3.3 Robots with Three Wheels

Since a robot with three wheels is statically stable and has a simple structure, it is one of the most widely used structures for wheeled robots. There are a large number of designs according to the choice of individual wheel types. Every wheel introduced in Sect. 24.2.1 can be used to construct three-wheel robots. In this section, five popular design examples are described (Fig. 24.7):

1. Two-wheel differential drive
2. Synchronous drive
3. An omnimobile robot with Swedish wheels
4. An omnimobile robot with active caster wheels
5. An omnidirectional robot with steerable wheels.

Two-Wheel Differential-Drive Robot

A two-wheel differential-drive robot is one of the most popular designs and is composed of two active fixed wheels and one passive caster wheel. The robot can be classified as a type (2,0) robot in the nomenclature of Sect. 24.2.6. It is possible to extend the robot to a four-wheel robot by adding passive caster wheels. The major advantages of the robot can be summarized as follows:

- A simple mechanical structure, a simple kinematic model, and low fabrication cost.
- A zero turning radius is available. For a cylindrical robot, the obstacle-free space can easily be computed by expanding obstacle boundaries by the robot radius r.
- Systematic errors are easy to calibrate.

On the other hand, its drawbacks are:

- Difficulty of moving irregular surfaces. When the robot goes over uneven surfaces, its orientation

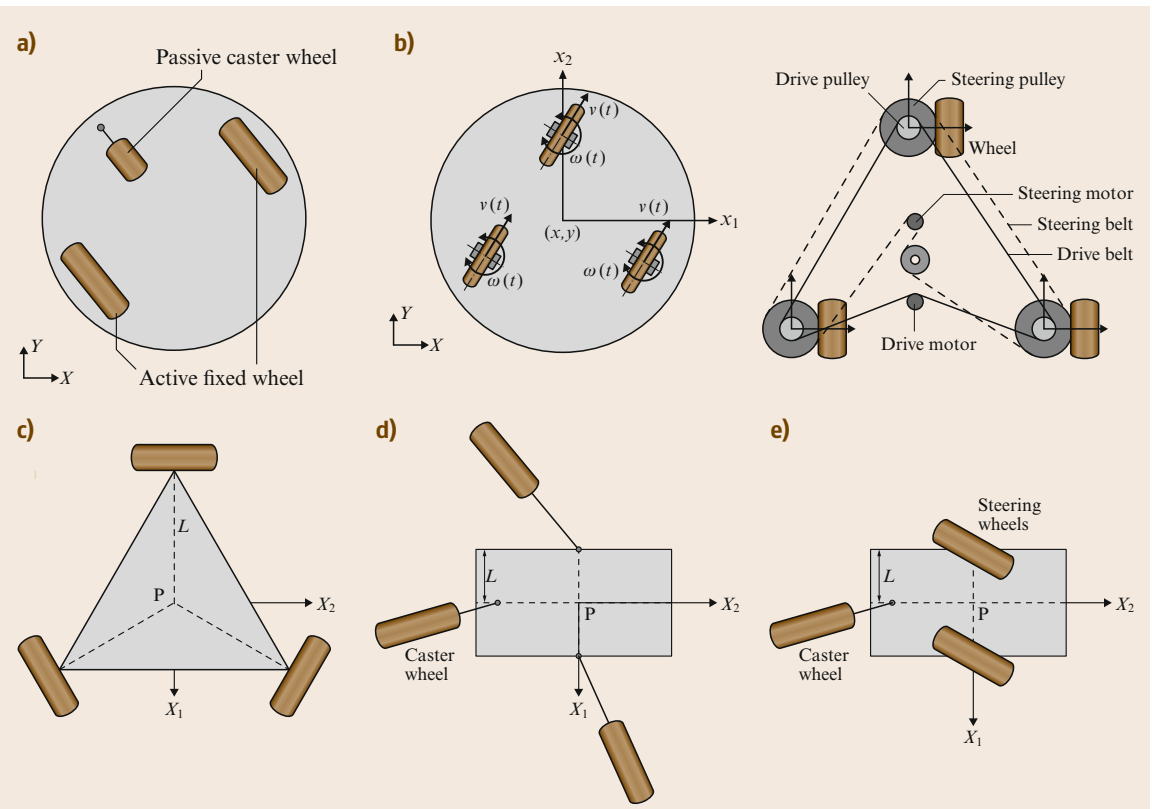


Fig. 24.7 (a) Two-wheel differential drive, (b) synchronous drive, (c) omnimobile robot with Swedish wheels, (d) omnimobile robot with active caster wheels, and (e) omnidirectional robot with active steerable wheels

might change abruptly if one of the active wheels loses contact with the ground.
- Only bidirectional movement is available.

Synchronous-Drive Robot

A synchronous-drive robot can be built by using centered or off-centered orientable wheels. The steering and driving motions of each wheel are mechanically coupled by chains or belts, and the motions are actuated synchronously, so the wheel orientations are always identical. The kinematic model of a synchronous drive robot is equivalent to that of the unicycle, a type (1,1) robot. Therefore, omnidirectional motion, i. e., motion in any direction can be achieved by steering the wheel orientations to the desired velocity direction. However, the orientation of the robot chassis cannot be changed. Sometimes a turret is employed to change the body orientation. The most significant advantage of the synchronous-drive robot is that omnidirectional movement can be achieved by using only two actuators. Since the mechanical structure guarantees synchronous steering and driving motions, less control effort is required for motion control. Other advantages include that odometry information is relatively accurate and driving forces are evenly distributed among all the wheels. The drawbacks of this approach can be summarized as:

- Complicated mechanical structure.
- If backlash or loose coupling is present in the chain transmission, velocity differences between wheels may occur.
- In order to achieve omnidirectional movement, the wheel orientations should be aligned to the desired velocity direction before movement, due to the nonholonomic velocity constraints.

Omnimobile Robot with Swedish Wheels

The omnimobile robot with Swedish wheels corresponds to type (3,0) in the nomenclature of Sect. 24.2.6. At least three Swedish wheels are required to build a holonomic omnidirectional robot. A major advantage of using the Swedish wheel is that omnidirectional mobile robots can be easily constructed. At least three Swedish wheels are required to build a holonomic omnidirectional robot. Since omnidirectional robots can be built without using active steering of wheel modules, the mechanical structures of actuating parts can have simple structures. However, the mechanical design of a wheel becomes slightly complicated. One drawback of the Swedish wheel is that there is a vertical vibration because of discontinuous contacts during motion. In order to solve this problem, a variety of mechanical designs have been proposed; examples can be found in [24.10] and [24.11]. Another drawback is its relatively low durability when compared to conventional tires. An example of a robot using Swedish wheels can be found in [24.12].

Omnimobile Robot with Active Caster Wheels

A holonomic omnidirectional robot can be constructed by using at least two active caster wheels, and the robot also belongs to type (3,0). The robot can be controlled to generate arbitrary linear and angular velocities regardless of the wheel orientations. Since the robot uses conventional tires, the disadvantages of Swedish wheels, for example, vertical vibrations or durability problems, can be solved. An example can be found in [24.13]. The disadvantages of this robot can be summarized as follows:

- Since the location of the ground contact point (i. e., footprint) changes with respect to the robot chassis, instability can take place when the distance between the wheels is too short.
- If the robot switches its movement to the reverse direction, an abrupt change of wheel orientations may take place. This is called the shopping-cart effect, which may result in instantaneous high steering velocities.
- If a driving motor is directly attached to the wheel, wires to the motor will be wound due to steering motions. In order to avoid this, a gear train should be employed to transmit the input angular velocity from the driving motor, which is attached to the robot chassis. In this case, the mechanical structure becomes quite complicated.
- If a robot is equipped with more than two active caster wheel modules, more than four actuators are used. Since the minimum number of actuator to achieve holonomic omnidirectional motion is three, this is an overactuated system. Therefore, actuators should be accurately controlled in a synchronous way.

Omnidirectional Robot with Active Steerable Wheels

Centered orientable wheels are also employed to build omnidirectional robots; at least two modules are required. A significant difference between the active caster wheel and the centered orientable wheel is that the wheel orientation should always be aligned with the desired direction of velocity direction, as computed by inverse kinematics. This fact implies that this robot is nonholonomic and omnidirectional: it is a type (1,2) robot. The control problem is addressed in [24.14]. The mechanical drawbacks are similar to those of using active caster wheels (i. e., many actuators and compli-

cated mechanical structures). Since the driving motor is directly attached to the driving axis in many cases, allowable steering angles are limited in order to prevent wiring problems.

There are a lot of design candidates for three-wheel robots, other than the five designs described earlier. They can be classified and analyzed according to the scheme presented in Sect. 24.1. The aforementioned designs can be extended to four-wheel robots to improve stability. Additional wheels can be passive wheels without adding additional kinematic constraints. Active wheels can also be added and should be controlled by solving the inverse kinematics problem. Four-wheeled robots require suspension to maintain contact with the ground to prevent wheels from floating on irregular surfaces.

24.3.4 Robots with Four Wheels

Among the various four-wheel robots, we focus on the car-like structure. The car-like structure has been called as the Ackermann steering geometry that was shown in Fig. 24.3c. The front two wheels should be synchronously steered to keep the same instantaneous center of rotation. It is clear that the orientations of two front wheels are slightly different because the curvatures of rotation are different. As a result, this solution is kinematically equivalent to a single orientable wheel and the robot can be classified as a type (1,1) robot. A major advantage of a car-like robot is that it is stable during high-speed motion. However, it requires a slightly complicated steering mechanism. If the rear wheels are actuated, a differential gear is required to obtain pure rolling of the rear wheels during the turning motion. If the steering angle of the front wheel cannot reach 90°, the turning radius becomes nonzero. Therefore, parking motion control in a cluttered environment becomes difficult.

24.3.5 Special Applications of Wheeled Robots

Articulated Robots

A robot can be extended to an articulated robot, which is composed of a robot and trailers. A typical example is the luggage-transporting trailer system at airports. By exploiting trailers, a mobile robot obtains various practical advantages. For example, modular and reconfigurable robots can change their configuration according to service tasks. A common design is a car with multiple passive trailers, which is the simplest design of an articulated robot. VIDEO 326 shows another example of a trailer robot. From the viewpoint of control, some significant issues have been made clear, including

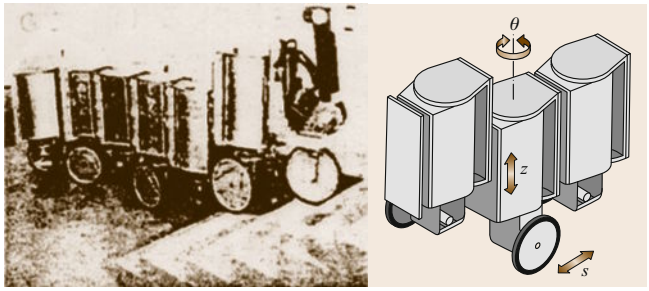


Fig. 24.8 An active trailer system (after [24.16])

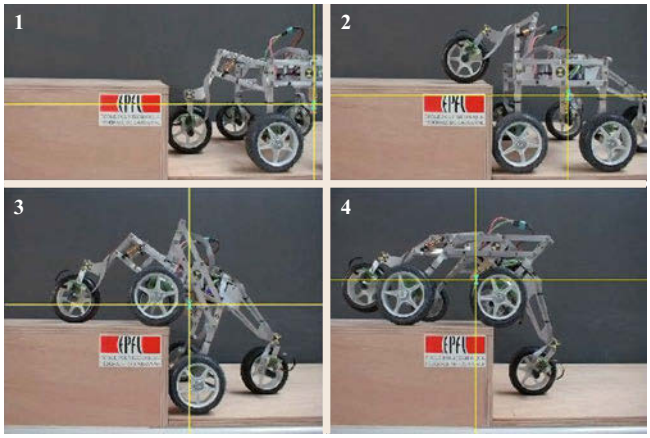


Fig. 24.9 A mobile robot for rough terrain (after [24.17])

a proof of controllability and the development of open- and closed-loop controllers using canonical forms such as the chained form. The design issues for trailer systems are the selection of wheel types and decisions regarding the link parameters. In practical applications, it is advantageous if trailers can move along the path of the towing robot. Passive trailers can follow the path of a towing robot within a small error by using a special design of passive steering mechanism for trailers; see, for example, [24.15].

On the other hand, active trailers can be used. There are two types of active trailers. A first approach is to actuate wheels of trailers. The connecting joints are passive, and two-wheel differential-drive robots can be used as active trailers. By using this type of active trailers, accurate path-following control can be achieved. The second approach is to actuate connecting joints. The wheels of the trailer are passively driven. By appropriate actuation of the connecting joints, the robot can move without wheel actuation, by snake-like motions. As an alternative design, we can use an active prismatic joint to connect trailers, in order to lift the neighboring trailer. By allowing vertical motion, a trailer system can climb stairs and traverse rough terrain. Examples of active trailers can be found in [24.16].

Avidbots Ex1016
Page 625 of 2154

Hybrid Robots

A fundamental difficulty of using wheels is that they can only be used on flat surfaces. To overcome this problem, wheels are often attached to a special link mechanism. Each wheel is equipped with independent actuators and a linkage mechanism enables the robot to adapt its configuration to irregular ground conditions (VIDEO 329). A typical design can be found in [24.17] and can be understood as a hybrid robot that is a combination of a legged robot and a wheeled robot. Another hybrid example is a robot equipped with both tracks and wheels. Wheels and tracks have complementary advantages and disadvantages. Wheeled robots are energy efficient; however, tracked robots can traverse rough terrain. Therefore, a hybrid robot can selectively choose its driving mechanism according to environmental conditions, although fabrication cost increases.

24.4 Wheel–Terrain Interaction Models

A wheeled robot's mobility properties are governed by forces generated at the wheel–terrain contact interface. The ability to accurately model wheel–terrain interaction forces is therefore an important aspect of robot design, simulation, and control. These forces are strongly influenced by the relative stiffness of the wheel and terrain. Generally, there are four possible wheel–terrain interaction cases. The first case is that of a rigid wheel traveling on rigid terrain (Fig. 24.10a). The second case is that of a rigid wheel traveling on deformable terrain (Fig. 24.10b). The third case is that of a deformable wheel traveling on deformable terrain (Fig. 24.10c). The fourth case is that of a deformable wheel traveling on rigid terrain (Fig. 24.10d). Models for these four cases are presented as follows. It should be noted that while many different types of models (i. e., finite element, discrete element, empirical) have been developed for each of these four cases, the focus here is on analytical models that are broadly suitable for analysis, simulation, design, and control purposes.

24.4.1 Rigid Wheels on Rigid Terrain

Many robots employ wheels made of metal, stiff rubber, or other materials that deform very little when subject to loading experienced during operation. When operated on stiff surfaces such as indoor flooring, pavement, or stone, wheel–terrain interaction can be reasonably approximated as a point contact. An interaction model based on classical Coulomb friction can then be employed to describe bounds on available tractive forces, F_x, and lateral forces, F_y, as a function of the load on the wheel, W, for a robot with n wheels traveling on a surface with coefficient of friction μ

$$F_{xi} \le \mu_x W_i\,, \quad i = 1 \ldots n\,, \tag{24.11}$$

$$F_{yi} \le \mu_y W_i\,, \quad i = 1 \ldots n\,. \tag{24.12}$$

Since the frictional force can be generated in any direction, and its magnitude is limited, a bound on the norm of the frictional and lateral forces can be expressed as

$$\left(\frac{F_{xi}}{\mu_x W_i}\right)^2 + \left(\frac{F_{yi}}{\mu_y W_i}\right)^2 = 1\,, \quad i = 1 \ldots n\,. \tag{24.13}$$

Equation (24.13) represents a concept known as the friction ellipse (Fig. 24.11). When the effective friction is equal in all directions the ellipse becomes a circle.

24.4.2 Rigid Wheels on Deformable Terrain

Robotic locomotion in outdoor, off-road terrain frequently results in deformation of the terrain, especially when the terrain is composed of a deformable material such as sand, silt, loam, or clay. When the wheel is constructed of a rigid material such as metal or stiff rubber, or the tire inflation pressure is high enough such that tire deformation is small, wheel–terrain interaction occurs along an arc of the wheel, rather than at a single point (Fig. 24.10b). In such scenarios, the Coulomb friction model described in Sect. 24.4.1 does not accurately represent the relationship between the wheel load

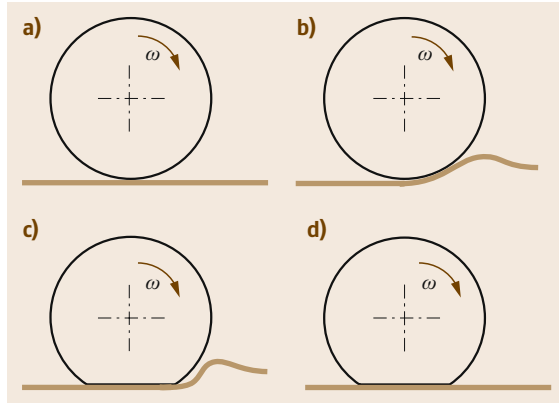


Fig.24.10a–d Four cases of wheel–terrain interaction mechanics: **(a)** rigid wheel traveling over rigid terrain, **(b)** rigid wheel traveling over deformable terrain, **(c)** deformable wheel traveling over deformable terrain, and **(d)** deformable wheel traveling over rigid terrain

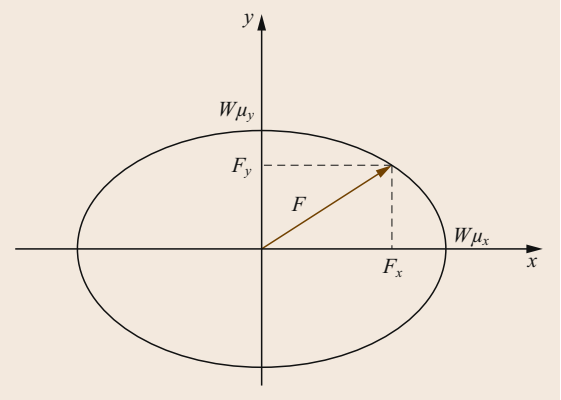


Fig. 24.11 Illustration of friction ellipse concept to represent constraint on longitudinal and lateral friction

and tractive forces. This is because the mechanism for force generation derives primarily from shearing along failure planes in the terrain, rather than from frictional contact at a point on the wheel–terrain interface.

Models of the interaction of rigid wheels on deformable surfaces were developed by *Bekker* in the 1950s and 1960s in the context of large military vehicles [24.18]. Bekker's research gave rise to the discipline of *terramechanics*, i. e., the study of vehicle-terrain interaction phenomena. Bekker's terramechanics models have subsequently been applied to robotic systems at a range of sizes [24.19] and [24.20]. However, Bekker performed his original analyses in the context of large military vehicles, which were typically several meters in length and with gross vehicle weights exceeding 500 kg. The accuracy of such models when used to analyze the performance of small robotic systems remains an open research question [24.21] and [24.22].

A description of terramechanics models for the case of rigid wheels on deformable surfaces is presented in Chap. 55, in the context of Space Robotics. However, Chap. 55 assumes that the wheel is constructed of a rigid material such as metal, which is typical for planetary exploration rovers but rare for robots operating on Earth. As noted earlier, a pneumatic tire will behave as a rigid wheel if the average tire ground pressure, which is a function of the tire carcass stiffness and inflation pressure, exceeds some critical ground pressure. The average tire ground pressure can be obtained experimentally by measuring the tire contact patch area while the tire is subjected to a series of known vertical loads and inflation pressures. The critical ground pressure is defined as

$$p_{\mathrm{gcr}} = \left(\frac{k_{\mathrm{c}}}{b} + k_{\phi}\right)^{\frac{1}{(2n+1)}} \left(\frac{3W}{(3-n)b_{\mathrm{ti}}\sqrt{D}}\right)^{\frac{2n}{2n+1}}, \tag{24.14}$$

where k_{c}, k_{ϕ}, n are the soil-dependent parameters, b is the smallest dimension of the (typically rectangular) wheel–terrain contact patch, b_{ti} is the tire width, W is the vertical load on the tire, and D is the tire diameter.

If the average tire ground pressure exceeds the critical ground pressure, a pneumatic tire can be modeled as a rigid wheel interacting with a deformable surface. This scenario is governed by the equations presented in Chap. 55. If the average tire ground pressure does not exceed the critical ground pressure, a pneumatic tire can be modeled either as a deformable wheel interacting with deformable terrain, or as a deformable wheel interacting with rigid terrain. Models of these scenarios are presented in Sects. 24.4.3 and 24.4.4.

24.4.3 Deformable Tires on Deformable Terrain

When the average tire ground pressure does not exceed the critical ground pressure, both the tire and the terrain surface may undergo deformation. Accurate modeling of this interaction thus requires consideration of phenomena related to both soil deformation (most notably, soil compaction, and shear deformation) and tire deformation. Several semiempirical models for this scenario, based on Bekker theory, have been developed in recent years.

A simple approach to modeling in this interaction scenario was initially proposed by Bekker, and further developed by *Harnisch* [24.23]. This approach relies on the observation that tire deformation serves to increase the size of the wheel–terrain contact patch, and thus reduce the average ground pressure and minimize sinkage. Bekker proposed the *substitute circle* concept, which embodies the notion that the deformable tire can be equivalently represented by a rigid tire of increased diameter, to yield identical sinkage (Fig. 24.12). The diameter of the larger rigid tire can be computed via the following method:

1. Assuming zero slip, sinkage for the original tire diameter D, assumed to be rigid, is calculated for the vertical load W and soil parameters.
2. Knowing the tire load-deflection characteristics, for the given inflation pressure the tire deflection δ can be calculated.
3. The diameter of the larger substitute circle D' will correspond to the diameter of the circle that passes through P_1, P_2, and P_3.

Once the outer circle diameter is known, all other quantities will be calculated following the standard methodology presented in Chap. 55, substituting D' for the original wheel diameter D.

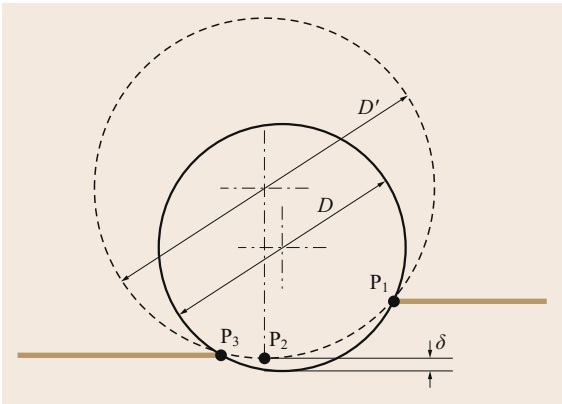


Fig. 24.12 Illustration of substitute circle concept

An alternative approach, based on a more physically realistic model of the deformed tire geometry, was proposed by *Wong* [24.24]. Here, the tire geometry is represented as a circular arc with a perfectly flat (i. e., horizontal) base (Fig. 24.13). The pressure distribution under the flat section BC is assumed to be uniform, and equal to the average ground pressure of the tire p_g. Disregarding shear effects and soil elastic rebound (section CD), it is possible to calculate the tire deflection δ by balancing vertical forces

$$W = bp_g l_t + W_{AB} \,, \tag{24.15}$$

where b is the tire width, l_t is the length of the flat section BC, and W_{AB} is the vertical reaction exerted by the soil along the undeformed tire section AB

$$W_{AB} = \left[b \left(\frac{k}{l} + k_\phi \right) \sqrt{D} (z_0 + \delta)^{n-1} \right] \frac{(3-n)(z_0+\delta)^{\frac{3}{2}} - (3-n)\delta^{\frac{3}{2}} - 3z_0\sqrt{\delta}}{3} \,, \tag{24.16}$$

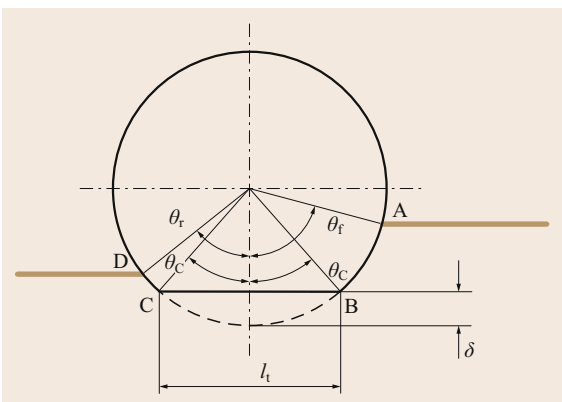


Fig. 24.13 Diagram of deformable wheel traveling over deformable terrain

where D is the undeformed tire diameter and z_0 is the static sinkage that can be calculated through standard pressure–sinkage relations

$$z_0 = \begin{cases} \left(\frac{p_g}{(k_c/l_t)+k_\phi} \right)^{\frac{1}{n}} & (l_t < b) \\ \left(\frac{p_g}{(k_c/b)+k_\phi} \right)^{\frac{1}{n}} & (l_t \geq b) \end{cases} \,, \tag{24.17}$$

where $l_t = 2\sqrt{D\delta - \delta^2}$ is the contact patch length. Although not strictly correct, the tire ground pressure p_g can be approximated by the tire inflation pressure, a known variable. Following this assumption, it is possible to solve (24.16) for the tire deflection δ.

Once the deformed tire shape has been calculated, all other relevant quantities (i. e., sinkage, drawbar force, and torque) can be calculated. Since the tire profile is nonsmooth, the normal and tangential stresses are described by piecewise expressions. The vertical load balance can be written as

$$W = br \int_{\theta_r}^{\theta_c} (\sigma \cos\theta - \tau \sin\theta) \mathrm{d}\theta + bDp_g \sin\theta_c + br \int_{\theta_c}^{\theta_f} (\sigma \cos\theta + \tau \sin\theta) \mathrm{d}\theta \,. \tag{24.18}$$

The aforementioned equation can be solved for the entry angle θ_f. The angle θ_c can be derived from the tire deflected shape. Once the tire-terrain geometry is determined, the sinkage can be found. Note that this new sinkage will differ from the static sinkage z_0. It is then possible to calculate drawbar pull F_x and driving torque

$$F_x = br \int_{\theta_r}^{\theta_c} (\sigma \cos\theta + \tau \sin\theta) \mathrm{d}\theta + b \int_0^{l_t} \tau \mathrm{d}x + br \int_{\theta_c}^{\theta_f} (-\sigma \cos\theta + \tau \sin\theta) \mathrm{d}\theta \,, \tag{24.19}$$

$$T = br^2 \int_{\theta_r}^{\theta_c} \tau \mathrm{d}\theta + br \cos\theta_c \int_0^{l_t} \tau \mathrm{d}x + br^2 \int_{\theta_c}^{\theta_f} \tau \mathrm{d}\theta \,. \tag{24.20}$$

Another flexible tire model developed by *Senatore* and *Sandu* is similar to the approach proposed by Wong, however it calculates the tire deflection as a function of inflation pressure, and also models multi-pass effects [24.25]. Generally speaking, all of the flexible tire models proposed to date have various advantages and drawbacks, and none have gained universal approval in the research community.

24.4.4 Deformable Tires on Rigid Terrain

In scenarios where the tire stiffness is significantly lower than the terrain stiffness, only the tire will experience significant deformation. This scenario has been studied extensively in the automotive community, since it is the typical interaction scenario for passenger and commercial vehicles operating on paved roads. Here, two common models of steady-state tire–terrain interaction are briefly described: the brush model and the magic formula model [24.26].

The brush model models the tire contact patch as a row of elastic bristles in contact with the ground. Carcass, belt, and tread element compliance is captured by a lumped bristle compliance. For the case of pure longitudinal slip, the tractive force can be calculated as

$$F_x = 2c_{px}a^2\frac{V_{sx}}{V_x}\,, \tag{24.21}$$

where c_{px} is the longitudinal tread element stiffness per unit length and other quantities are illustrated in Fig. 24.14.

In a similar manner, the lateral force can be calculated as follows

$$F_y = 2c_{py}a^2\alpha\,, \tag{24.22}$$

where α is the slip angle. As previously discussed for the case of rigid wheels on rigid terrain, combined slip operation will limit the availability of longitudinal and lateral thrust [24.26].

The magic formula model is an empirical model that has found widespread application due to its flexibility and reduced computational burden. The model is based on a combination of trigonometric functions that generate a curve that (usually) passes through the origin, reaches a peak value, and then tends to a horizontal asymptote: this behavior is typical of force/moment versus slip characteristics of modern tires, regardless of tire size, construction, inflation pressure, and other characteristics. The general form of the magic formula model is

$$y = D\sin\{C\arctan[Bx - E(Bx - \arctan Bx)]\}\,, \tag{24.23}$$

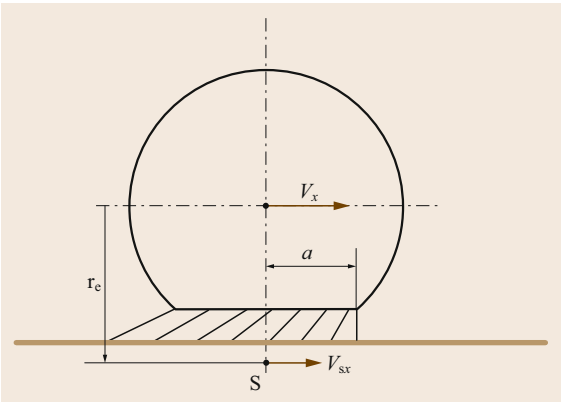


Fig. 24.14 Brush tire model for analyzing interaction of deformable tire and rigid terrain

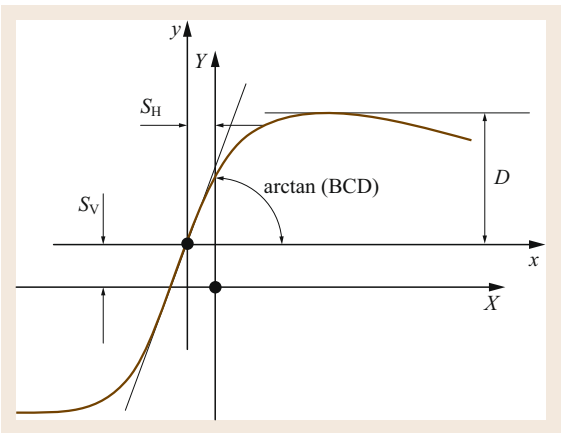


Fig. 24.15 Diagram illustrating influence of parameters of magic tire model

with $Y(X) = y(x) + S_V$ and $x = X + S_H$. Here, Y represents the output variable, which may be F_x, F_y or M_z, and X represents the input variable, which may be $\tan\alpha$ or k (i. e., the lateral or longitudinal slip). The relation also contains several empirically determined values, as follows: B represents a stiffness factor; C a shape factor; D a peak value; E a curvature factor; S_H a horizontal shift; and S_V a vertical shift. These factors are illustrated in Fig. 24.15.

24.5 Wheeled Robot Suspensions

A suspension is a system of linkages, springs, dampers, and actuators that govern the relative motion between a robot's wheels and body. These mechanisms include a degree of freedom allowing the wheel to spin, and optional degrees of freedom for steering and vertical translation. Suspension mechanisms are useful when driving over uneven surfaces for several reasons. First, a suspension allows the wheels to maintain contact with the ground despite disturbances from the uneven ground surface, which allows generation of traction, braking, and cornering forces. Additionally, suspensions provide load-bearing and dissipative elements (i. e., springs,

dampers, and active or semiactive actuators) to mitigate the effect of disturbances on robot body motion, and thus allow for improved obstacle negotiation and sensor or payload stabilization. Key issues in suspension design include the design of suspension linkage kinematics and selection of load-bearing and dissipative elements.

Suspension linkage kinematics are designed to allow vertical displacement of the wheels. Typical suspension mechanisms include simple prismatic joints, rotational joints with a wheel attached to a trailing arm, and 4-bar linkages. For car-like wheeled robots, substantial inspiration can be drawn from the design of passenger vehicle suspensions, which has received vast research attention [24.27]. A comprehensive treatment of the design of suspension linkage kinematics is beyond the scope of this chapter. Interested readers are referred to [24.28].

The selection of load-bearing and dissipative elements is typically based on analysis of low-order models of the suspension subject to expected operational conditions. Common methods for modeling passive and semiactive suspensions are presented in the following. Analysis of these models allows for principled selection of springs, dampers, and active or semiactive actuators that form the basis of nearly all wheeled robot suspensions.

24.5.1 Passive Suspension Dynamics

Arguably the most common passive suspension design consists of a parallel spring and damper mounted between the robot body and wheels. A simple model for analyzing the vertical dynamics of this suspension is commonly referred to as the quarter car model [24.29] and is illustrated in Fig. 24.16. The fraction of body mass supported by the suspension element, m_b, is connected to a wheel with mass m_w by a spring element with stiffness k_s and a damping element with damping coefficient b_s. The wheel stiffness is denoted by k_w, and transmits vertical excitation from the uneven terrain surface. The heights of the body mass, wheel mass, and terrain surface are given by z_b, z_w, and z_u, respectively.

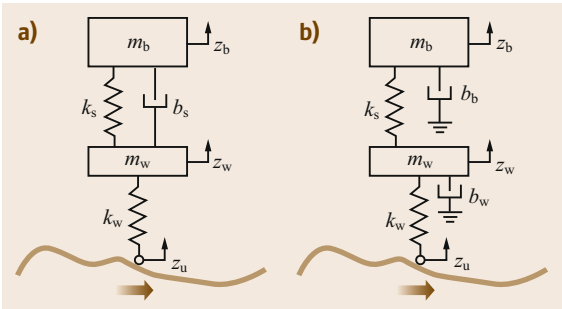


Fig.24.16a,b Quarter car model for passive suspension modelling **(a)** and quarter car model with ideal sky-hook semi-active suspension control **(b)**

A state vector x for this model is composed of the suspension displacement $d_s = z_b - z_w$, wheel spring displacement $d_w = z_w - z_u$, and velocities $\dot{z}_b$, $\dot{z}_w$, as shown in (24.24), along with a linear state space model (24.25)–(24.27). Note that the input to the model is the vertical terrain velocity $\dot{z}_u$. This is a convenient formulation for linear systems analysis since $\dot{z}_u$ can be modeled as white noise with intensity proportional to the vehicle's forward speed and a road roughness parameter [24.29].

$$x = \begin{pmatrix} d_s \\ d_w \\ \dot{z}_b \\ \dot{z}_w \end{pmatrix} \tag{24.24}$$

$$\dot{\boldsymbol{x}} = \mathbf{A}\boldsymbol{x} + \mathbf{B}\boldsymbol{u} \tag{24.25}$$

$$\mathbf{A} = \begin{pmatrix} 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 1 \\ -k_s/m_b & 0 & -b_s/m_b & b_s/m_b \\ k_s/m_w & -k_w/m_w & b_s/m_w & -b_s/m_w \end{pmatrix} \tag{24.26}$$

$$\mathbf{B} = \begin{pmatrix} 0 \\ -1 \\ 0 \\ 0 \end{pmatrix} . \tag{24.27}$$

To investigate the properties of this model, transfer functions $G(s) = \mathbf{Y}(s)/\mathbf{U}(s)$ are computed for outputs of interest $\boldsymbol{y} = \mathbf{C}\boldsymbol{x}$ as

$$G(s) = \mathbf{C}(s\mathbf{I} - \mathbf{A})^{-1}\mathbf{B} . \tag{24.28}$$

With a white noise input, all frequencies are equally represented. Thus, the magnitude of the transfer function $|G(j\omega)|$ at a given frequency ω represents the frequency response of each output. Two outputs are of particular interest for suspension design purposes: the wheel displacement d_w and body acceleration $\ddot{z}_b$. This is because the wheel displacement corresponds to variations in the wheel contact force, and thus informs the road holding and maneuvering capability of the robot. The body acceleration represents the amount of vibration experienced by the sensors and payloads mounted on the robot body.

The output matrix for the wheel displacement $\mathrm{d}w$ is

$$\mathbf{C} = (0 \quad 1 \quad 0 \quad 0) , \tag{24.29}$$

and the output matrix for the body acceleration $\ddot{z}_b$ is

$$\mathbf{C} = \left(-\tfrac{k_s}{m_b} \quad 0 \quad -\tfrac{b_s}{m_b} \quad \tfrac{b_s}{m_b} \right) . \tag{24.30}$$

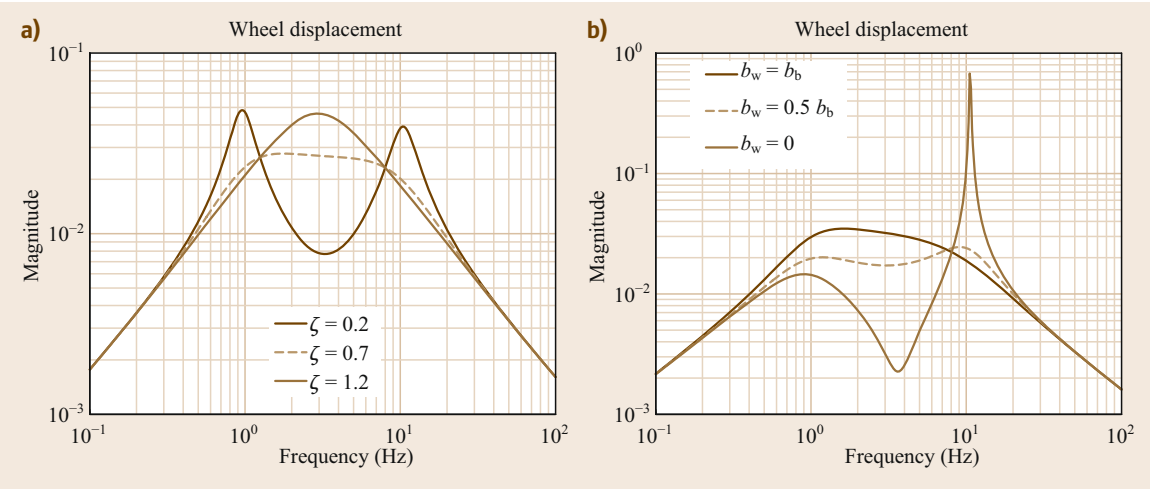


Fig.24.17a,b Wheel displacement frequency response for passive suspension (a) and semiactive suspension. Wheel displacement frequency corresponds to wheel contact force variation and road holding capability

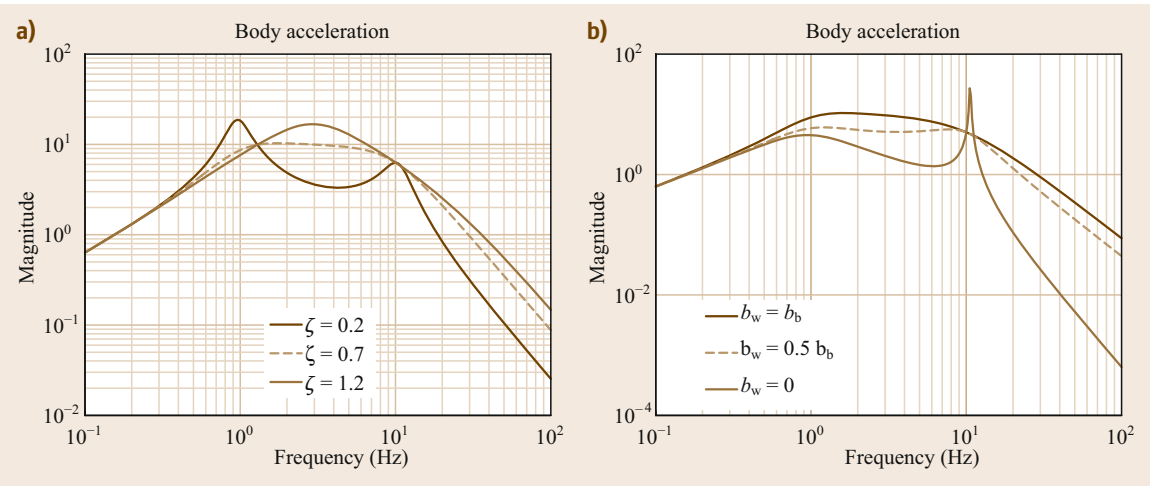


Fig.24.18a,b Body acceleration frequency response for passive suspension (a) and semiactive suspension (b). Body acceleration frequency corresponds to the vibration of sensors and payloads mounted on the robot body

The frequency response of wheel displacement and body acceleration are shown in Figs 24.17 and 24.18, respectively. These plots are shown with the following representative parameter values for a passenger vehicle: the ride frequency $\omega_b = \sqrt{k_s/m_b} = 2\,\pi$ rad/s, the wheel hop frequency $\omega_w = \sqrt{k_w/m_w} = 20\,\pi$ rad/s, the mass ratio $\rho = m_w/m_b = 0.1$, and the damping ratio $\zeta = b_s/(2\sqrt{k_s m_b}) = (0.2, 0.7, 1.2)$. Various damping ratios are shown to illustrate the behavior of underdamped, critically damped, and over-damped suspensions.

For the underdamped case, where $\zeta = 0.2$, resonant peaks (which are generally undesirable) exist in the frequency response near ω_b and ω_w for both the wheel displacement and body acceleration. For the over-damped case, where $\zeta = 1.2$, the peak response occurs between the resonant peaks. For the (approximately) critically damped case, where $\zeta = 0.7$, the peak responses are reduced, and for this reason many suspension are designed to exhibit critical damping.

24.5.2 Semiactive Suspension Dynamics

The introduction of controllable actuators to the suspension allows for shaping of the system's dynamic response, and can improve road holding performance

and reduce body vibration. Suspension systems integrated with controllable actuators are referred to as active or semiactive suspensions. Active suspensions refer to systems with powered rotational or linear actuators capable of both supplying and absorbing significant amounts of energy, such that the actuator effectively replaces the passive components in a traditional suspension. Such systems are rare due to their complexity and cost. A simpler and more inexpensive (and therefore more common) approach to controllable suspensions is to employ semiactive elements. This refers to elements that can absorb energy in a controlled manner, such as active (e.g., magnetorheological) fluid dampers. Semiactive suspensions can be modeled in a manner similar to passive suspensions with a time-varying, positive damping coefficient $b_s(t) \geq 0$.

Semiactive suspension control schemes attempt to control the damping coefficient $b_s(t)$ to emulate dampers with coefficients b, b acting on the absolute velocities of the body and wheels, in order to reduce the effect of wheel vibration on the robot body. A common technique known as a sky-hook controller is illustrated in Fig. 24.16. The technique can be approximated by applying the damper force F given in the following. Note that a semiactive system can only approximate this control, as F is constrained as $F(\dot{z} - \dot{z}) \leq 0$.

$$F = -b\dot{z} + b\dot{z} \tag{24.31}$$

To illustrate the potential improvement achievable by the use of a semiactive suspension with a sky-hook controller, frequency responses of the semiactive suspension quarter car model with F given earlier are plotted in Figs 24.17 and 24.18. The model parameters are identical to the values from the previous section for critical damping $\zeta = 0.7$, with damping coefficients $b = b$ and $b = (b, 0.5b, 0)$. Note that the case $b = b$ is identical to the passive, critically damped case.

With no wheel damping, $b = 0$, both frequency responses are improved for $\omega \ll \omega$, and the body acceleration response is improved for $\omega \gg \omega$. Near ω, however, there exist substantial resonant peaks for both outputs. With moderate wheel damping $b = 0.5b$, the amplitude of the resonant peaks near ω are reduced and both peak responses are reduced, though performance improvements of $b = 0$ are not matched for frequencies far from ω.

While the sky-hook and other suspension control algorithms can improve the dynamic response of the vehicle, it is important to note some inherent trade-offs in semiactive and active suspension control. It has been proven that road holding and attenuation of body vibration can be concurrently improved through active control for frequencies below the wheel hop frequency, ω [24.29]. At frequencies above ω, however, any attempts to improve one of these quantities necessarily causes the other to degrade.

24.6 Conclusions

Wheels are the most commonly employed running gear for mobile robots due to their relative simplicity, robustness, and low cost. The number of possible wheeled mobile robot realizations is almost infinite, depending on the number, type, implementation, geometric characteristics, and motorization of the wheels. This chapter has described several such realizations. Notwithstanding this variety, it is possible to classify WMRs into only five generic categories. This categorization aids understanding of wheel structures through simplification. Practical robot structures have been classified according to the number and type of wheels. Wheel-terrain interaction models have been presented to allow the analysis of tractive force generation capability, for design and simulation purposes. Finally, a brief description of common suspension systems has been presented, including a presentation of structures and dynamic models.

Video-References

VIDEO 325 An omnidirectional mobile robot with active caster wheels
available from http://handbookofrobotics.org/view-chapter/24/videodetails/325

VIDEO 326 Articulated robot – a robot pushing 3 passive trailers
available from http://handbookofrobotics.org/view-chapter/24/videodetails/326

VIDEO 327 An omnidirectional robot with 4 Mecanum Wheels
available from http://handbookofrobotics.org/view-chapter/24/videodetails/327

VIDEO 328 An omnidirectional robot with 4 Swedish wheels
available from http://handbookofrobotics.org/view-chapter/24/videodetails/328

VIDEO 329 An innovative space rover with extended climbing abilities
available from http://handbookofrobotics.org/view-chapter/24/videodetails/239

References

24.1 H. Asama, M. Sato, L. Bogoni: Development of an omnidirectional mobile robot with 3 DOF decoupling drive mechanism, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (1995) pp. 1925–1930

24.2 L. Ferriere, G. Campion, B. Raucent: ROLLMOBS, a new drive system for omnimobile robots, Robotica **19**, 1–9 (2001)

24.3 W. Chung: *Nonholonomic Manipulators, Springer Tracts Adv. Robotics*, Vol. 13 (Springer, Berlin, Heidelberg 2004)

24.4 J.E. Colgate, M. Peshkin, W. Wannasuphoprasit: Nonholonomic haptic display, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (1996) pp. 539–544

24.5 G. Campion, G. Bastin, B. dAndrea-Novel: Structural properties and classification of kinematic and dynamic models of wheeled mobile robots, IEEE Trans. Robotics Autom. **12**, 47–62 (1996)

24.6 R. Nakajima, T. Tsubouchi, S. Yuta, E. Koyanagi: A development of a new mechanism of an autonomous unicycle, Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS) (1997) pp. 906–912

24.7 G.C. Nandy, X. Yangsheng: Dynamic model of a gyroscopic wheel, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (1998) pp. 2683–2688

24.8 Y. Ha, S. Yuta: Trajectory tracking control for navigation of self-contained mobile inverse pendulum, Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS) (1994) pp. 1875–1882

24.9 Y. Takahashi, T. Takagaki, J. Kishi, Y. Ishii: Back and forward moving scheme of front wheel raising for inverse pendulum control wheel chair robot, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (2001) pp. 3189–3194

24.10 K.-S. Byun, S.-J. Kim, J.-B. Song: Design of continuous alternate wheels for omnidirectional mobile robots, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (2001) pp. 767–772

24.11 M. West, H. Asada: Design and control of ball wheel omnidirectional vehicles, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (1995) pp. 1931–1938

24.12 B. Carlisle: An omnidirectional mobile robot. In: *Development in Robotics*, ed. by B. Rooks (IFS, Bedford 1983) pp. 79–87

24.13 M. Wada, S. Mori: Holonomic and omnidirectional vehicle with conventional tires, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (1996) pp. 3671–3676

24.14 D.B. Reister, M.A. Unseren: Position and constraint force control of a vehicle with two or more steerable drive wheels, IEEE Trans. Robotics Autom. **9**(6), 723–731 (1993)

24.15 Y. Nakamura, H. Ezaki, Y. Tan, W. Chung: Design of steering mechanism and control of nonholonomic trailer systems, IEEE Trans. Robotics Autom. **17**(3), 367–374 (2001)

24.16 S. Hirose: *Biologically Inspired Robots: Snake-Like Locomotion and Manipulation* (Oxford Univ. Press, Oxford 1993)

24.17 R. Siegwart, P. Lamon, T. Estier, M. Lauria, R. Piguet: Innovative design for wheeled locomotion in rough terrain, J. Robotics Auton. Syst. **40**, 151–162 (2003)

24.18 M.G. Bekker: *Introduction to Terrain-Vehicle Systems* (Univ. Michigan Press, Ann Arbor 1969)

24.19 H. Shibly, K. Iagnemma, S. Dubowsky: An equivalent soil mechanics formulation for rigid wheels in deformable terrain, with application to planetary exploration rovers, J. Terramech. **42**, 1–13 (2005)

24.20 G. Ishigami, A. Miwa, K. Nagatani, K. Yoshida: Terramechanics-based for steering maneuver of planetary exploration rovers on loose soil, J. Field Robotics **24**(3), 233–250 (2007)

24.21 G. Meirion-Griffith, M. Spenko: A modified pressure-sinkage model for small, rigid wheels on deformable terrains, J. Terramech. **48**(2), 149–155 (2011)

24.22 C. Senatore, M. Wulfmeier, P. Jayakumar, J. Maclennan, K. Iagnemma: Investigation of stress and failure in granular soils for lightweight robotic vehicle applications, Proc. Ground Vehicle Syst. Eng. Technol. Symp. (2012)

24.23 C. Harnisch, B. Lach, R. Jakobs, M. Troulis, O. Nehls: A new tyre–soil interaction model for vehicle simulation on deformable ground, Vehicle Syst. Dyn. **43**(1), 384–394 (2005)

24.24 J.Y. Wong: *Theory of Ground Vehicles*, 3rd edn. (Wiley, Hoboken 2001)

24.25 C. Senatore, C. Sandu: Off-road tire modeling and the multi-pass effect for vehicle dynamics simulation, J. Terramech. **48**(4), 265–276 (2011)

24.26 H.B. Pacejka: *Tire and Vehicle Dynamics*, 2nd edn. (Elsevier, Oxford 2005)

24.27 P. Barak: *Magic Numbers in Design of Suspensions for Passenger Cars*, SAE Tech. Pap. No. 911921 (SAE, Warrendale 1991)

24.28 J.C. Dixon: *Suspension Geometry and Computation* (Wiley, Chichester 2009)

24.29 J.K. Hedrick, T. Butsuen: Invariant properties of automotive suspensions, Proc. Inst. Mech. Eng. Part D J. Automob, Eng. **204**(1), 21–27 (1990)

Avidbots Ex1016
Page 633 of 2154

Multimedia Contents



25. Underwater Robots

Hyun-Taek Choi, Junku Yuh

Covering about two-thirds of the earth, the ocean is an enormous system that dominates processes on the Earth and has abundant living and nonliving resources, such as fish and subsea gas and oil. Therefore, it has a great effect on our lives on land, and the importance of the ocean for the future existence of all human beings cannot be overemphasized. However, we have not been able to explore the full depths of the ocean and do not fully understand the complex processes of the ocean. Having said that, underwater robots including remotely operated vehicles (ROVs) and autonomous underwater vehicles (AUVs) have received much attention since they can be an effective tool to explore the ocean and efficiently utilize the ocean resources. This chapter focuses on design issues of underwater robots including major subsystems such as mechanical systems, power sources, actuators and sensors, computers and communications, software architecture, and manipulators while Chap. 51 covers modeling and control of underwater robots.

25.1 **Background** 595
25.2 **Mechanical Systems** 596
25.2.1 Frames and Pressure Hulls 596
25.2.2 Materials and Buoyancy 597
25.3 **Power Systems** 599
25.3.1 On-Board Power Sources 599
25.3.2 Surface Power Systems 601
25.4 **Underwater Actuators and Sensors** 601
25.4.1 Actuators 602
25.4.2 Nonacoustic Sensors 602
25.4.3 Acoustics Sensors 604
25.4.4 Other Sensor Issues 606
25.5 **Computers, Communications, and Architecture** 606
25.5.1 On-Board Computers 606
25.5.2 Communications 608
25.5.3 Software Architecture 610
25.6 **Underwater Manipulators** 614
25.6.1 Dynamics 614
25.6.2 Teleoperation 615
25.6.3 Autonomous Manipulation 616
25.7 **Conclusions and Further Reading** 617
Video-References 617
References 618

25.1 Background

The ocean covers about two-thirds of the earth and has a great effect on the future existence of all human beings. Underwater robots or unmanned underwater vehicles (UUVs) can help us better understand marine and other environmental issues, protect the ocean resources of the earth from pollution, and efficiently utilize them for human welfare.

Most UUVs commercially available in the market are tethered and remotely operated, referred to as remotely operated vehicles (ROVs). The operators on the mother vessel control the ROV by sending control signals and power to the vehicle and receiving video, status, and other sensory data via an umbilical cable. Most work-class ROVs are equipped with one or multiple video cameras and lights, sonars, a still camera, one manipulator, one grabber, and a wide range of sampling devices. The large deep-sea ROVs use a tether management system (TMS) that allows the vehicle to be deployed using a strong but heavy umbilical cable and then flown out from the TMS using a lighter, more flexible cable. ROVs must have high bandwidth for high-resolution video and data transmission.

Avidbots Ex1016
Page 634 of 2154

The US Navy funded the early ROV technology development in the 1960s. They were used to recover underwater ordnance. ROVs have been used for various applications including military operations, environmental systems, scientific missions, ocean mining, and gas and oil industry [25.1]. Visiongains analysis indicates that total spending on ROVs in the energy industry will reach about $2.7 billion in 2022 as ROVs are required for more inspection, maintenance, drill support, and decommissioning activities than ever before.

However, since most commercial ROV systems currently have some operational issues such as very high operational costs with mother vessels and operator fatigue, the demand for advanced underwater robot technologies that would lead to next generation workclass ROV systems or autonomous underwater vehicles (AUVs) has been growing. A survey paper on AUVs was published by *Yuh* [25.2] in 2000. The paper describes key areas, such as dynamics, control systems, navigation, sensors, communications, power systems, pressure hulls, fairing, and mechanical manipulators. It also presents existing AUV systems and components commercially available in the market then.

This chapter focuses on design issues with each major subsystem of underwater robots, such as mechanical systems, power sources, actuators and sensors, computers and communications, software architecture, and manipulators.

25.2 Mechanical Systems

Every 10 m adds approximately 1 atm of water pressure on a robot body and the chemical environment of the seawater is highly corrosive. Therefore, mechanical systems that have excellent structural strength, and excellent chemical and environmental resistance are needed.

An underwater robot's mechanical structure can be divided into three functional subsystems: frame, pressure vessel, and fairing. Frame is the primary skeleton of the robot. It defines the robot's shape, supports its weight both in and out of the water, provides mechanical support, resists hydrodynamic forces of motion and additional force due to operation of tools, and protects against any damages [25.3]. Its design must be based on structural analysis. A pressure vessel should be a waterproof structure and it must be strong enough to withstand water pressure at a target design depth. Typically, a cylindrical or spherical shape is the most effective way to maximize strength at the expense of low space efficiency. A pressure vessel with a larger diameter would result in increased cost of materials and machining since it requires increasing the thickness of the pressure vessel for the same strength. Consequently, the pressure vessel would become heavier. Therefore, selecting appropriate materials for a pressure vessel is very important. Fairing is like a skin of the robot. A fairing that covers all or parts of an underwater robot offers some advantages, such as an aesthetically pleasing exterior, a decreased drag force, and protection.

25.2.1 Frames and Pressure Hulls

The frame must be designed considering the following operational conditions:

1. On the ground without motion
2. Underwater in motion
3. Hanging from a crane for launch and recovery.

The major points of acting force for each condition are totally different. The static gravitational force is enough to be considered for the first case. For the second case, we should consider buoyancy force, gravitational force as well as all thruster forces in a dynamic state. During the launch and recovery stages, the enormous snap force, due to the mother ship's vertical motion can be concentrated on a lifting point of the robot.

The structural design and analysis must be conducted to take advantages of shapes offering high strength and stiffness for their weight. At the same time, materials should be chosen to give the maximum structural performance with the minimum weight [25.4]. An important point is that weights in both air and water must be carefully calculated. One should note that buoyancy materials will also increase the overall weight of the robot in air. A factor of safety is one of crucial parameters in designing mechanical systems. Incorporating a factor of safety may result in 50% overdesign for a manned system and 10% overdesign for an unmanned system [25.5]. Figure 25.1 shows a deep-sea ROV, HEMIRE's open frame that is made of aluminum. HEMIRE was developed by Korea Research Institute of Ships and Ocean Engineering (KRISO) (VIDEO 796). On the other hand, a closed frame or a fairing is desirable for AUVs, due to limited on-board power sources for required hydrodynamic performance.

The first and ultimate goal in designing a pressure vessel is to provide a watertight environment, where on-board computers, sensors, or normal batteries are installed. Therefore, two important design factors are wa-

terproof sealing design and water pressure-resistant design. When given specifications, such as an inside diameter (ID) or an outer diameter (OD), length, thickness, and materials, we must analyze buckling stress and yield stress. They are the dominant factors for the deep-sea (high pressure) environment [25.6]. A spherical- or cylindrical-shaped pressure vessel would be good to withstand the hydrostatic pressure; however, a pressure vessel of cylindrical shape is considered a better alternative as it provides a better placement for the components. Hydrodynamic design would not be needed for pressure vessels placed inside a frame with fairing. However, if the body of an underwater robot is designed as a pressure vessel, hydrodynamics in designing a pressure vessel must be considered to reduce the drag, cavitation, and acoustic noise [25.7].

Frames and pressure vessels must have the structural analysis using the finite element method (FEM) software, and nondestructive tests and/or pressure testing to avoid any mechanical defects. Pressure testing is the best way to reveal hidden defects in materials and in machining process even though it requires additional time, cost and nontrivial facilities. It is a critical environmental test because a hydrostatic pressure test, cyclic pressure test for fatigue test due to repeated deployments, and failure mode test to compare a designed crush depth with its real depth are all involved [25.5]. Pressure testing facilities provide not only a chamber for external pressure testing but also various types of measurement sensors and systems to record any changes during the testing.

25.2.2 Materials and Buoyancy

Besides high strength and high stiffness, one should consider good corrosion resistance and machinability in choosing materials for frames and pressure vessels [25.7].



Fig. 25.1 Aluminum frame of an ROV, Hamire, developed by KRISO, Korea

A typical underwater robot for shallow water tends to use aluminum, polycarbonate (PC), and polyoxymethylene (POM), while high strength steel, titanium, and composites are considered for a deep-sea underwater robot. The hybrid ROV, *Nereus*, is for a depth of 11 000 m and employs two pressure vessels composed of 96% alumina ceramic because its high compressive strength-to-weight ratio allows for near neutrally buoyant pressure vessels capable of going to these extreme depths [25.8]. Basic properties of materials are described later.

Steel is commonly used because of its high tensile strength and low cost. For marine applications, surface treatment of steel must be done because it is susceptible to corrosion or rusting. It tends to distort magnetic fields. Type 316 stainless steel is usually regarded as the standard marine grade and it is typically used for hydraulic, pneumatic and fasteners, but it is difficult to machine and weld. The aluminum alloy is lightweight and has high strength. It is of reasonable value and not related to magnetic effects. Since it is vulnerable to corrosion, surface treatment is also required. High heat conductivity is another advantage for a pressure vessel of aluminum alloy. The titanium alloy has a very high strength-to-weight ratio, excellent corrosion resistance without any surface treatment, low electric conductivity, and no magnetic field distortion. Drawbacks include low heat conductance, difficulties in machining and welding, and high cost.

The most commonly used composite for marine applications is glass-fiber reinforced plastic (GFRP), which has a very high strength-to-weight ratio, excellent corrosion resistance, and a competitive price tag. Carbon fiber reinforced plastic (CFRP) performs better, but it is more expensive. For deep-sea applications, CFRP would bring lots of advantages. The pressure vessel of the Seaglider for 6000 m was made of CFRP because of its ability to achieve weight to displacement ratios of less than 0.5 [25.9]. However, it has not been popular in the market mainly because of difficulties in machining. Metal matrix composites (MMC) and ceramic alloy have many advantages over other materials. In particular, they have high elastic modulus and hardness, high melting points, low thermal expansion, and good chemical resistance.

Instead of metals and composite materials, acrylic (plexiglas), POM (polyoxymethylene, acetal, delrin), PC (polycarbonate lexan), ABS (acrylonitrile–butadiene–styrene), PVC (polyvinyl chloride), and plastic are alternatives for underwater robots operating in shallow-water environments. The main advantages of the abovementioned materials are excellent

Table 25.1 Properties of materials for an underwater robot

Specification	Density (kg/dm^3) (typical)	Tensile strength (MPa), Ultimate (typical)/Yield (typical)	Modulus of elasticity (GPa) (typical)
Steel (HY80)	7.87	482–689/276–517	190–210 (200)
Stainless steel (Type316)	7.75–8.1	620–560/415–290	193
Aluminum 6061-T6	2.7	310/276	68.9
Aluminum 7075-T6	2.81	572/503	71.7
Titanium Ti-6Al-4V	4.43	950/880	113.8
GFRP, E-glass	2.44–2.48	4800–4900/NA	85–90
GFRP, S-glass	2.54–2.60	3400–3500/NA	65–75
CFRP	1.75–1.95	2500–6000/NA	200–800
Acrylic	0.7–1.25 (1.16)	19.3–85.0 (64.3)/25.0–85.0 (60.2)	0.950–3.79 (2.69)
Polycarbonate	0.95–1.54 (1.2)	46.1–124 (64.6)/37.0–191 (64.3)	1.80–7.58 (2.37)
Unreinforced polyoxymethylene	1.14–2.23 (1.42)	5.00–115 (55.6)/22.0–120 (26.8)	0.586–12.1 (2.73)
Acrylonitrile–butadiene–styrene	0.35–3.5 (1.06)	24.1–73.1 (38)/20.0–73.1 (43.3)	0.778–6.10 (2.3)
Polyvinyl chloride	1.4	4.00–59.0/17.0–52.0	0.00159–3.24
Ceramic (96% alumina)	3.7–3.97	2070–2620/260–300	393
Metal matrix composite, Al/SiC	2.78–3.01	230–317	125–270

corrosion resistance and ease of machining and fabrication with relatively low cost. Additionally, POM has high stiffness, low friction, and excellent dimensional stability. PC and acrylic have high-impact strength. They are lightweight and flexible. Even though glass is brittle and, hence, subject to damage from impact, glass spheres are also appropriate for pressure vessels because of the immense strength-to-weight ratio, corrosion resistance, nonmagnetic and electrically nonconductive characteristics, and low cost [25.9]. Some properties are summarized in Table 25.1.

The metal corrosion in seawater is the gradual deterioration of a material based on the electrochemical reaction. There are galvanic corrosion and electrolytic corrosion which are known for fundamentally the same process and known for accelerating rates of the metal corrosion in underwater robot [25.3]. To protect against electrolytic corrosion, insulators, such as resin or rubber are attached to metal contact surfaces [25.6]. However, surface treatment is an effective way to avoid any direct contact of seawater. Anodizing is a surface conversion process that changes the exterior physical properties of a metal to form a hard coating on the surface as results of a controlled electrochemical process. One thing we should remember is that anodizing works well only when the entire surface remains intact. In other words, a small chip in the surface can become the point of accelerated corrosion if there is contact with a less active material [25.10]. A topcoat will provide additional protection against corrosion. Sacrificial anodes, made of highly active metal like zinc and magnesium, are alternative for corrosion protection. As sacrificial anodes corrode and deteriorate, they must be periodically inspected and replaced if needed.

When designing an underwater robot, it is recommended to have small positive buoyancy, except the reserve buoyancy for payload. It will help an underwater robot returning to the surface in the case of any electronics and power failure. The amount of positive buoyancy would depend on the size of robots, typically from a few hundred grams to several kilograms. There are several factors for consideration in choosing buoyancy materials, such as low density, high strength, nontoxic harmless, and weather resistance. The most important factors are low water absorption and low compressibility, which are all dependent on the target operating depth [25.4]. For an underwater robot, expandable polystyrene, polyisocyanurate foam, and syntactic foam are the most common buoyancy materials. The expandable polystyrene (EPS) is a super light and cheap material. However, it is only used for very shallow water because it is compressed by water pressure. Polyisocyanurate foams are generally low-density, insulation-grade foams, which have excellent insulating value and good compressive-strength properties. Since it is usually made in large blocks via a continuous extrusion process, these blocks are reasonably inexpensive and effective for shallow water [25.4]. The syntactic foam consists of glass microballoons and epoxy resin. The amount of trapped air within the resin structure will determine the density as well as the durability of the foam at deeper depths. It is also noted here that ballasting is a procedure of adding, removing, or relocating weights or floatation on an underwater robot to correct its buoyancy, pitch, and roll. There are two types of ballast systems: fixed (static) ballast and variable (active) ballast. In a fixed ballast system, an underwater robot is preset to the desired ballast and operates without any dynamic change of the ballast system. Fixed payload is

usually in the form of several lead blocks, which can be exchanged for equipment without adjusting the robots buoyancy. Variable ballast is not common in most ROVs but the hybrid ROV, *Nereus*, has one [25.8].

25.3 Power Systems

Underwater robots obtain power from any of three categories: on-board power, surface power by a tether, and the combination of the two. For an onboard power system, we can think of many types of power sources, such as primary/secondary batteries, internal/external combustion engines, and fuel cells. For surface powered underwater robots, the mother ship has to deliver power to the robot by a tether. Surface powered underwater robots have advantages of high power capacity and almost unlimited power but also entail additional cost for a winch, a drum, a TMS, and a mother ship. This section reviews typical power sources including batteries, fuel cells, and surface power systems.

25.3.1 On-Board Power Sources

The on-board power source of an underwater robot is an air-independent power (AIP) system, and the most common AIP system is batteries. Performance specifications of untethered vehicles like AUVs, such as working range, service time, and payloads are limited by the battery capacity.

Battery technology has improved over the last decade. One would like to have better battery sources in terms of high energy density, high charge and discharge rates, desirable current characteristics, wide operating/storage temperature range, reliability, long lifecycle, low cost for purchasing and maintenance, and, most importantly, safety against fire and explosion. We should also consider details on safety such as nongassing, spill-proof electrolyte, status monitoring as well as special requirements depending on the shape and volume of an AUV and particular working environment conditions such as depth, under sea ice near the North Pole [25.6, 11, 13].

Primary batteries usually have better endurance than secondary batteries. The lithium primary battery has a very high energy density but its operating cost is more expensive. The alkaline battery can be considered the most common primary battery as it is relatively cheap, but it has safety issues because it outgases hydrogen and it is sensitive to temperature [25.7].

There are a number of different kinds of secondary batteries including lead-acid, silver zinc, nickel–cadmium, nickel-metal hydride, lithium-ion, lithium-polymer batteries. Lead-acid batteries are the oldest type of battery having a low energy density. However, because of its relatively large power rate and low cost, they are the popular choice for many applications. Nickel–cadmium (Ni–Cd) batteries were widely used in commercial products because of their robust characteristics but had problems with the toxicity of the heavy metal cadmium. Nickel metal hydride batteries have been improved to eliminate drawbacks of Ni–Cd and replaced Ni–Cd in the market. During the last couple of decades, the silver–zinc battery was the most used power source in professional AUVs because of its better performance compared to that of old type secondary batteries such as lead-acid and Ni–Cd, even though it is very expensive. Today, lithium-ion and lithium-polymer batteries are popular in our daily life. We can find them in many IT devices. Li-ion batteries using lithium ions as an electrolyte have the advantages of higher energy density, longer cycle life, relatively low self-discharge, low maintenance, and light weight. However, the drawbacks include protection circuits and aging effects. In contrast, lithium-polymer batteries use a solid polymer composite. This reduces the chance of overcharging and electrolyte leakage and is more robust against physical damage [25.12], even though they have a lower energy density and a lower life cycle compared to Li-ion batteries. Some properties are summarized in Table 25.2.

In the literature, as one of the alternatives, seawater batteries with very long endurance capabilities are considered and have been studied to improve the performance. However, due to low power capacity, its applications are limited [25.14].

A fuel cell is an electrochemical device converting the chemical energy of fuels with oxygen directly into electricity to produce electric power, heat, and water. Since there is continuously replenished fuel, a fuel cell has a much larger capacity and longer endurance than batteries at the cost of system complexity. There are various types of fuel cells. The most common types are solid oxide (SOFC), direct methanol (DMFC), proton exchange membrane (PEMFC, or polymer electrolyte (PEFC)), phosphoric acid (PAFC), molten carbonate (MCFC), and alkaline (AFC), where each type operates at different temperature conditions using different materials [25.15, 16].

A closed-cycle PEFC system is known to be the most suitable for underwater and it consists of a fuel cell generator, a high pressure oxygen tank, and a metal hydride tank. Its operating temperature is around 60 °C

Table 25.2 Summary of characteristics of commonly used batteries (after [25.7, 11, 12])

Type	Advantages	Disadvantages	Energy/weight (W h/kg)	Energy/volume (W h/l)	Rating	Life cycle
Alkaline (primary)	Cheap	Leak gas at high temperature	140	–	–	–
Lithium (primary)	High energy density, expensive		375	–	–	–
Lead acid (secondary)	Low cost, long life cycle (1000)	Low energy density, leak hydrogen	20–30	60–80	−40–50	700
Silver–zinc (Ag–Zn) (secondary)	High energy density	Limited life cycle (40–100), need careful maintenance, high cost	100–120	180–200	−48–71	100
Nickel–cadmium (Ni–Cd) (secondary)	Long life cycle, flat discharge curve, good low temperature performance	Memory effects, environmentally unfriendly, exothermic charging process	40–60	50–150	−40–60	2000
Nickel–metal hydride (Ni–MH) (secondary)	High energy density, reduced memory effect	Self-discharging, limited discharge current	60–120	140–300	Ambient	1000
Lithium ion (secondary)	High-energy density, long life cycle, no memory effect	Aging effect	100–265	250–730	−20–45	400–1200
Lithium polymer (secondary)	Flexible form factor, light weight, Improved safety	Lower energy density compared to Li-ion	130–200	170–300	Ambient	300–3000

and its reactive product is only pure water. Urashima, the first prototype of a fuel cell-driven underwater vehicle developed by JAMSTEC, Japan, uses a solid polymer electrolyte fuel cell (PEFC) which allows 60 h endurance and 300 km cruising with max. 4 kW power, and they achieved 60 h of nonstop electric power generation and finished a sea trial [25.17]. In 2007, JAMSTEC started the research to develop a fuel cell for the second generation long-range cruising AUV (LCAUV) to cruise over 3000 km with over 600 h of continuous running [25.12].

The energy density of a fuel cell generator itself is high, but when we consider a whole fuel cell system including subsystems, such as hydrogen, oxygen, and reactant water tanks, auxiliary components, and control electronics, its energy density becomes low. Downsizing of these subsystems is needed for underwater applications. So, a hybrid fuel/battery system would be an alternative to share full loads because it allows both components to be of smaller dimensions and operates with higher efficiency using direct hydrogen and pure oxygen [25.18].

There are two types of loads: propulsion load for propulsion and hotel load for on-board computers, lights, navigation sensors, and payloads. If an AUV is operated for professional long-term missions, we need very accurate simulations for adjusting trade-offs among various constraints. Conventional lead-acid and Ni-based batteries would not be the best choice for AUVs. Instead, lithium-based batteries are the attractive solution that does not entail time-consuming estimation techniques. Nonetheless, lead-acid batteries remain a practical and low-cost option for education, experimentation, and testing.

There are two options for battery installation inside an AUV. One is that they are housed in a pressure vessel, the other one is using pressure-tolerant and waterproof batteries without a pressure vessel. The latter eliminates burdens related to leakage in assembling the system, thus increases system reliability. Lithium-polymer batteries using the solid electrolyte were made without air in the packaging. Hence, they would be a candidate in designing pressure-tolerant on-board power source for AUVs [25.19].

A battery management system (BMS) is a very important subsystem for performance and safety. It monitors the internal status of voltages, current, temperatures and state of charge. Based on the above information, a BMS controls circuits for balancing and protects batteries from over-current, over/under-voltage, over/under-temperature, and any predescribed conditions. As a part of health monitoring, an on-board computer of an AUV should collect status from the BMS to reflect a power budget and health to given missions.

Just like adding volume of auxiliary systems and fuel storage units to calculate an energy density of a fuel cell, we need a measure for fair comparison of the energy density among different power sources. Since a power source is installed inside of an AUV, which should be neutrally buoyant to save the energy in operation, energy density per neutrally buoyant

Table 25.3 Summary of pros and cons of DC and AC (alternating current) (data from [25.4, 20])

Form	Pro	Con	Remarks
DC	Two conductors with less shielding in tether, low voltage drop	Current protection, difficulties in voltage step-up and down	For observation class ROV
AC	Longer transmission distances, flexibility in voltage up and down	Inductance noise, reactive losses, need subsystem for freq. converting	For deep-sea ROV

kilogram should be a reasonable unit. This means that weight and volume for buoyancy materials and pressure vessels must be added to the power source itself, and this also means that the characteristics of buoyancy materials and pressure vessels (aluminum, titanium, composite material) under considering working (or maximum) depth should be designed before or at the same time [25.21]. Between batteries and fuel cells, a semi-fuel cell can be another alternative. It is a generator, but its usability is like a battery because one reactant is stored externally and brought to the reaction cell [25.11]. Aluminum/hydrogen peroxide energy semi-fuel cells can practically generate an energy density of about 400 Wh/kg (theoretically 3418 Wh/kg) which is three times that of a lithium-ion system [25.12]. HUGIN 3000 powered by a semi-fuel cell developed by FFI (Norwegian Defense Research Establishment) [25.7] operates at 4 km down to a sea depth of 3000 m for up to 60 h. This pressure-tolerant system is very attractive in deep-water systems. But it requires trained personnel for operation and maintenance as well as special facilities and tools [25.14].

25.3.2 Surface Power Systems

In the case of ROVs, the whole power is delivered from a mother ship to the ROV through an umbilical/tether cable, offering very high power and theoretically unlimited endurance. These electrical power transmission techniques may look simple, but actually involve a highly complicated procedure for optimization. A power system determines work capability of ROV as well as operating cost, subsystem size, and safety considerations. Typically, it consists of a surface system, a tether, and a subsea system. A surface system includes a shipboard power source (or power outlet), frequency converter, step-up transformers, and a ground fault monitoring device. A frequency converter is used only for high-frequency alternative current, but it is advantageous in handling the power of a subsea system. In a subsea system, there are step-down transformers, rectifiers, power busses, power switching devices for distribution, and a ground fault monitoring device. Also, system configuration and setup reflects the power requirement of the devices installed on a ROV including thrusters, lights, sensors and any circuits. Once again, these design parameters of a power system are not independent of each other [25.4, 20, 22, 23].

A detail configuration depends on the form of current and needed capacity. First of all, we have to make a decision on the form of current. There are three types of current for ROV systems:

1. 50/60 Hz
2. High frequency alternating current (HFAC) at 400 Hz
3. Direct current (DC).

The decision should be based on the design objectives of maximizing power transfer, efficiency and lifetime, as well as minimizing the overall system cost using steady-state analysis and transient analysis [25.20]. Table 25.3 shows pros and cons of the form of current.

Besides maximizing the power transfer to an ROV, we have to minimize the diameter of a cable, system cost, support systems on deck, and the weight of the ROV without losing a high level of electrical power quality. Therefore, most observation class ROVs use the DC power system mainly due to portability and cost, and work-class ROVs have a tendency to use a three phase AC power system.

25.4 Underwater Actuators and Sensors

It is inevitable that the overall performance of an underwater robot depends on the quality of its actuators and sensors. While one can think of different types of actuators for underwater robots, such as thrusters, water jets, and biomimetic actuators (VIDEO 793), most underwater robots use thrusters that use propellers involving hydrodynamics of water flow when generating force to move through water. It is noted that the mapping between thruster command and generated force is not linear, especially in the transient state of motion. The sensors for an underwater robot can be divided into three groups: navigation sensors, system sensors,

and mission sensors. They can also be categorized further by acoustic- or nonacoustic-based sensing. Sensing in underwater environments is very challenging, especially position sensing. There is no electromagnetic wave propagation under water except for very short ranges and, therefore, the GPS (global positioning system) is not available. Due to the limited performance of a single sensor, the sensor fusion approach with multiple sensors is used. For example, the inertial navigation system (INS) includes an IMU (inertial measurement unit) and a DVL (Doppler velocity logger) with highly complicated algorithms. The performance of optical sensing depends on the turbidity level, while acoustic sensing encounters significant challenges due to the dynamic properties and extreme nature of the ocean, such as multipath problems, variation of signal attenuation, variation of acoustic channel, nonlinear effects, and many more [25.24, 25]. This section describes underwater actuators and sensors, and their design-related issues.

25.4.1 Actuators

There are two major types of thrusters – electrical thrusters and electro-hydraulic thrusters – along with other types such as bio-inspired smart actuators and jet propulsion. The efficiency of an electro-hydraulic thruster is about 15%, while the efficiency of a fully electrical thruster is about 43% [25.13], where the definition of efficiency is the ratio of shaft mechanical power output versus electrical power input. In spite of the low energy efficiency, electro-hydraulic thrusters are preferred for large size work-class ROVs because of their extensive thrust and practicality in the rugged conditions of the offshore work environment. Various factors contributing to the power conversion efficiency of the hydraulic propulsion system for ROVs are described in [25.26].

Many small ROVs and AUVs have fully electrical thrusters because of their compact size and high efficiency. Torpedo-type AUVs for cruising commonly use the screw-type propellers, which generate much more power, though only in the forward direction. Small ROVs and AUVs requiring slow but fine motion typically use the propellers generating the same power in both forward/reverse directions. When two thrusters are operating on the same plane and the same direction of motion, an underwater robot must have counter-rotating thrusters to eliminate the counter torque effect of the thrusters rolling the robot opposite to the direction of propeller rotation.

There are many propeller design parameters including propeller diameter, the number of blades, blade pitch, blade thickness, hub size, rake, etc. Propeller diameter and blade pitch are critical design parameters that impact a robot's torque and power, and are related to the size and cost of an underwater robot. Therefore, we have to find an optimum combination of propeller diameter and blade pitch, along with careful matching to a motor to have an efficient electrical propulsion system [25.27]. A typical underwater electrical thruster includes an electric motor, a motor amplifier and controller, a thruster housing typically made of aluminum or titanium, and oil-filled, drive shafts designed for watertight using magnetic coupling or dynamic seal, propellers, and ducts. The Kort nozzle has a relatively simple geometry and it performs efficiently at low speed. The Rice nozzle has a lower drag coefficient and a better geometry to promote increased thrust [25.24].

25.4.2 Nonacoustic Sensors

Various nonacoustic sensors can be considered for measuring an underwater robot's position and pose, and detecting its linear and rotational motion. Magnetic sensors are the most common to get the north direction, but are susceptible to local magnetic disturbances generated by the active elements of a robot system or any magnetized materials in the environment.

Inertial sensors are sensors based on inertia, where linear acceleration is measured by accelerometers and the rate of rotation is measured by gyroscopes without an external reference. There are several types of accelerometers, such as pendulous accelerometers, pendulous integrating gyroscope accelerometers, force rebalance accelerometers, piezoelectric accelerometers, differential capacitor accelerometers, and vibrating beam accelerometers. They have each advantages and disadvantages. Some of them can be built using either conventional mechanical type or microelectromechanical system (MEMS) type. MEMS types like capacitive-based MEMS accelerometers and MEMS vibrating beam accelerometers, show generally higher sensitivity and better resolution [25.28, 29]. A gyroscope (gyro) is used to measure angular velocity of an object. There are three main types of gyro: spinning mass, vibratory, and optical. Spinning-mass type gyros operate on the principle of conservation of angular momentum, and this mechanical type gyro shows wide performance range (0.0001 to $> 100^\circ/\mathrm{h}$). However, it is very expensive. Vibratory (vibrating structure) type gyros use Coriolis forces that are small and of low cost because they use MEMS technology, but they have a limited performance range. Optical type gyros operate under the principle of the Sagnac effect, and there are two main types of optical gyros, RLG (ring-laser gyro) and interferometric fiber-optic gyro (IFOG). A RLG has high sensitivity

and stability, a wide range of measurement (from 0.001 to 1000°/h), excellent scale-factor stability and linearity, insensitivity to accelerations and fast turn-on (<1 s). The RLG technology is a mature technology, and is well established in the medium- and high-performance markets. However, RLGs are still complex and expensive. A fiber-optic gyro (FOG) can provide several important advantages of a pure solid-state device, such as reliable and low maintenance due to no moving part, no sealed cavity and no acoustic noise, and high performance. The FOG technologies are expected to replace many of the current RLG-based systems. However, one particular area where the RLG is expected to retain its superiority over the FOG is in applications requiring extremely high-scale-factor stability.

An attitude and heading reference system (AHRS) is a sensor providing three-dimensional (3-D) orientation information. Typically, it consists of (MEMS) gyroscopes, accelerometers, and magnetometers on all three axes, and an onboard microprocessor running a nonlinear estimation algorithm, such as an extended Kalman filter to compute the orientation. An integration drift is compensated by reference vectors, such as gravity and the earth magnetic field.

The fusion of several navigation sensors can minimize navigation errors. This sensor combination of accelerometers, gyros, navigation computer, and some electronics, called inertia navigation system (INS), provides estimations of velocity, position, and orientation by integrations of the linear accelerations and angular rates with an appropriate transformation between the body frame and the earth frame if given an initial position and orientation [25.12, 23]. Therefore, an INS is one of the most important devices for an underwater robot. Typically, the performance of an INS is usually rated in terms of its position error growth rate. USAF SNU84-1 categorizes INS units as shown in Table 25.4.

To increase the accuracy of a navigation system, an INS has been combined with DVL (Doppler velocity log). As an example, an integrated Doppler-inertial system, Marpos, calculates the relative position from a start point by dead reckoning, which shows a positioning accuracy of around 0.03% of the total distance traveled, the equivalent of 1.7 m per hour at a vehicle speed of three knots by AUV Maridan M600 [25.30].

Table 25.4 Performance classification of INS (USAF SNU84-1)

INS classification	Position error growth rate	Heading errors
Low	> 2 nm/h	> 0.2°
Medium	0.5–2 nm/h	0.05 to 0.2°
Precision	< 0.5 nm/h	< 0.05°

For the vertical motion of an underwater robot, its depth is easily measured by pressure sensors without any cumulative error. Actually, a pressure sensor measures the external pressure experienced by an underwater robot, where the two most common pressure sensor technologies are (i) strain gauges and (ii) quartz crystals. Strain gauge pressure sensors can typically attain overall accuracies of up to about 0.1% of full scale and resolutions of up to about 0.01% of full scale. Quartz crystal pressure sensors can typically attain overall accuracies of about 0.01% of full scale and overall resolution of up to about 0.0001% of full scale, that is, a resolution of one part per million [25.31]. Common units for pressure are atmosphere (atm), bar, kilopascals (kPa), pound of force per square inch (psi), Torr (= mmHg), and one atmosphere (1 atm) of pressure is 101.325 kPa and one bar (1 bar) of pressure is 100 kPa. In sea water, since pressure increases by 1 atm for every 10.0 m of depth, depth is an approximate reading of a pressure sensor multiplied by 10 m/atm. In fresh water, this simple formula underestimates the actual pressure by 2–3%, so you can get a more accurate depth by using 10.3 instead of 10 m/atm [25.3]. Measured pressure by a sensor is relative pressure. For absolute pressure, we have to add 1 atm to the sensor measurement.

Another nonacoustic sensor is an underwater computer vision system that can be considered as part of navigation sensors for underwater robots in addition to simply recording images of the underwater environment. When the underwater environment has high turbidity and insufficient and/or nonuniform light, it would not be easy to obtain enough visual information for the navigation through identifying objects, detecting obstacles, measuring distance between objects, and so on. However, when approaching its target very closely, for example, at the final stage of approaching for docking, the robot could gather useful information using a vision system.

On the other hand, underwater images are major products of underwater missions, especially for the inspection of submerged structures or observation of marine organisms, seafloor geology, etc. In this case, associating time stamp and localization data with the images increases the overall quality of the output. Because of the limited light range, it is very difficult to cover a wide area with one single shot. Thus, an image mosaicking is a popular strategy to overcome this in which several images are merged together in order to create a larger image, just as is done on Google Earth. Real-time image mosaicking is used for navigation stitches images with simple image manipulations, and high-quality image mosaicking is done by post-processing using various optimization techniques.

Avidbots Ex1016
Page 642 of 2154

More recently, advanced, high-quality camera and video management systems have been developed such as HD (high definition) cameras, HD-SDI (serial digital interface) transmission through a fiber-optic cable, high-quality digital video management systems, holographic cameras, laser scanning systems, 3-D camera systems with 3-D monitors, a hybrid optical-acoustic imaging systems incorporating stereographic optical cameras, and high-resolution multibeam sonars [25.12, 32, 33]. Such developments would greatly help increasing the effectiveness of the underwater robot operation.

25.4.3 Acoustics Sensors

There are many ways to calculate the speed of an underwater robot. We can simply estimate the speed by the integral of linear acceleration from an accelerometer even though there are large accumulative errors. We can also measure the speed directly using a pitot tube velocity log, a rotor type velocity log, an electromagnetic velocity log, and an acoustic Doppler current profilers (ADCPs) [25.6], which are all water speed sensors, and are measuring speed with respect to the water, not the ground speed of a robot. ADCP provides measurements of the water's current direction and magnitude by measuring the velocity of particles in water using principles of Doppler shift. However, Doppler velocity logs (DVLs) can be used to measure the velocity of robots moving in the water with respect to the ocean floor. Since ADCP and DVL use the same principles, typically DVL is operated in ADCP mode for a hydrographic instrument and bottom-lock mode for a navigation instrument. Since a DVL measures the Doppler shift of sonar signals reflected by the ground to obtain the velocity, there is a maximum range from seabed to the sensor, which is inversely proportional to the frequency of a signal, typically from 300 kHz to 2 MHz. Table 25.5 shows the performances of DVLs currently available in the market. Recently phased array technology in DVL has been announced and has brought significant advantages over piston array type DVL in terms of size and weight, which have extended the areas of applications for DVLs [25.34].

Table 25.5 Summary of DVL specifications available in market

Specification	Model A	Model B	Model C
Frequency	300 kHz	600 kHz	1200 kHz
Max. range	200–300 m	90–110 m	20–30 m
Velocity range	±10 m/s		
Velocity resolution	0.1 cm/s		
Accuracy	0.4% ± 2 mm/s	0.2% ± 1 mm/s	0.2% ± 1 mm/s
Average power	3–10 W	2–6 W	2–3 W

There are several types of sonar systems such as single-beam directional sonar, scanning sonar, side scan sonar, multibeam sonar, interferometric sonar and synthetic aperture sonar. Single-beam sonars send a pulse of sound at a particular frequency that is reflected from the seabed, and its echo is received by a transducer. It can be implemented in the form of a single channel, for example, an altimeter measuring the distance between a robot and seabed, or multiple channels, for example, obstacle avoidance sonar measuring distances between a robot and any objects around it. A scanning sonar, that is, forward looking sonar, has a small transducer rotating on its axis and emits pulses at the particular direction, and then obtains a set of bins, where a bin represents the echo intensity returning from a specific place along the transducer axis. An image is synthesized by placing the set of bins in a Cartesian space with a set of colors assigning their intensities. On the other hand, a side scan sonar has a fixed transducer emitting fan-shaped acoustic pulses in the cross-track direction, and then the measured acoustic reflections are compiled to create and accurate image of the sea bottom. It is one of the most effective tools for quickly searching for any objects such as wreckage debris or mines on a large area of the seabed. However, it does not provide reliable information on their position [25.35].

An imaging sonar uses a fan beam typically with a 30° 3 dB beam width while retaining about 1.7° angular resolution, whereas a profiling sonar emits a spot or pencil beam with about a 1.7° 3 dB beam width from both horizontal and angular rotation, which means that it provides higher resolution measurements. A multibeam sonar has an array of transducer emitting acoustic pulses simultaneously to cover a large area. This beam pattern is wide in the across-track field and narrow in the along-track field, which is appropriate for collecting highly accurate seafloor information at the cost of the high purchasing expense and high power consumption. An interferometric sonar is one possible tool providing wide-swath coverage and high-resolution bathymetry in shallow water using the technique of superimposing two (or more) waves, which significantly improves the efficiency and safety of hydrographic survey operations. A synthetic aperture sonar (SAS) is a high-resolution acoustic imaging technique that sums multiple returns from multiple pings to produce output imagery with constant along track resolution which is independent of both range and frequency. This is a kind of artificial enlargement of a sonar array by accurate navigation data. Dual frequency technology has been announced for an imaging sonar and side scan

sonar, which offers more flexibility. A typical dual frequency imaging sonar uses frequency between 0.9 and 2.5 MHz to obtain medium and ultra-high-quality resolution from mid- to close-range. A side scan sonar's typical range is from 100 to 500 kHz. Based on a high-quality imaging sonar, researchers have been trying to identify underwater objects. Figure 25.2 shows a result of real-time tracking for artificial landmarks using DIDSON [25.36] (VIDEO 794), where the multiple landmarks are being tracked after being detected and recognized by a probabilistic based algorithm developed by the Korea Research Institute of Ships and Ocean Engineering (KRISO) [25.37].

Acoustic positioning systems are the popular approach to locate the underwater robot by solving a triangulation problem based on distances between an underwater robot and the preinstalled transponders. The distances are calculated by the time of flight of the acoustic signal between them, that is, half the round-trip time multiplied by the speed of sound in water. It is theoretically simple with no cumulative error. However, to obtain accurate position information, we have to have an exact understanding of the system, consideration of the inherent physical error sources, and lots of practical experience. The accuracy inherently depends on the configuration of the baseline, and the speed of sound underwater is affected by water temperature, pressure, and salinity. Additionally, in shallow water or in the highly reverberant underwater environment, multiple reflected copies of any transmitted signal reach the receiver at delayed intervals. It is known as the multipath problem. Therefore, we have to know the characteristics of the underwater acoustic environment at the work site. Before explaining the configuration of types of acoustic positioning system, we need to review functions of its basic component. When an interrogate sends a signal, a transponder installed on the seabed or an underwater robot receives the interrogation signal, and then sends a reply signal to the interrogator. A responder is a transmitter installed on the seabed or an underwater robot just like a transponder, but it can be triggered by a hard-wired external control signal to transmit a signal. If a transmitter sends out a pulse on a particular frequency, it is called a beacon or pinger [25.4].

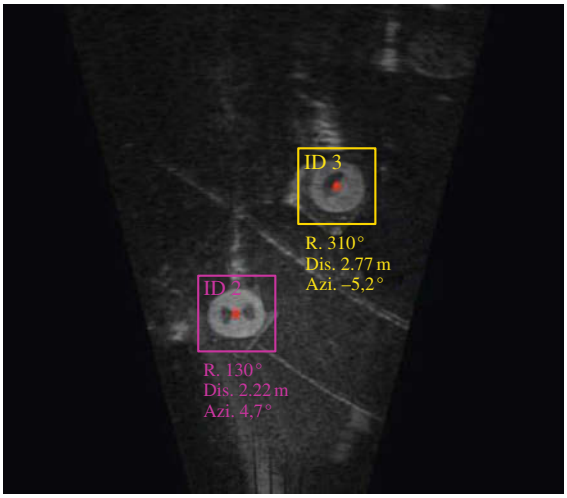


Fig. 25.2 Result of real-time tracking for artificial landmarks using DIDSON

There are three main types of acoustic positioning systems: long baseline (LBL), short baseline (SBL), and ultrashort baseline (USBL). An LBL system determines the position by measuring the slant ranges from three or more transponders mounted widely spaced on the seabed and typically at the corners of the operations site. This results in very high positioning accuracy and robustness regardless of water depth over wide operating area. However, this system requires multiple transponders which inevitably need a long deployment/recovery time and a calibration time required at each location. An update rate is relatively long because of two-way ranging. In an SBL system, there are three or more transducers connected by wires to a central control unit; these transducers construct a baseline installed on from a large barge to a small boat. It is just like an upside down version of an LBL system without requiring any seafloor premounted transponders. However, accuracy depends on the transducer spacing and mounting method. When a wider spacing is available, the performance can be similar to LBL systems. Otherwise, the accuracy goes down. A USBL system consists of a small integrated transducer array and a transponder/responder installed on what we want to trace, where the position of the transponder/responder is calculated by the combination of a target distance by the time of flight of a received signal and a target direction by the phase shift of the received signal on each of the acoustic elements in the transceiver array, called phase-differencing. Since a small error of the target direction calculated by a USBL will become a larger position error over a great distance, it should be mounted on the bottom end of a strong and rigid transducer pole, which is installed either on the side or in some cases on the bottom of a surface vessel. Additional sensors, including high-quality IMU and GPS, have to compensate for errors due to the any changing position and orientation of the integrated transducer array.

USBL systems offer many advantages, such as low complexity, ease of use and not requiring sea floor transponders. But they have also some disadvantages. An installation of a rigid pole for mounting of a transducer array is needed and a calibration process is required to determine offsets for the precise alignment

of a USBL system. Most importantly, their positioning accuracy and robustness are not as good as the LBL systems. In the case of range, all types are quite accurate assuming that the line of sight is maintained. However, the bearing accuracy of LBL systems is less than 1°, whereas SBL and USBL systems typically have 1–3° bearing accuracy [25.4].

The GPS intelligent buoys (GIB) system is available as a promising acoustic positioning system [25.38]. The configuration of GIB is an inverted LBL, where several floating buoys equipped with submerged hydrophones and GPS receivers measure the time of arrivals (TOAs) of the acoustic signals sent by the emitter of an underwater target and from the time measurements tracked position is calculated in real time. This system allows fast and easy installation without a calibration process and a dedicated mother ship and provides real-time multiobjects tracking with a fast update rate because of one-way acoustic signals from the emitter of an underwater target to the floating buoys. The number of floating buoys is determined by the size of the area and the accuracy requirement of given operations.

25.4.4 Other Sensor Issues

Many sensors and communication devices of an underwater robot, such as altimeter, DVL, imaging sonar, ranging sonar, acoustic modem, and acoustic positioning system are sonar-based devices. The integration and concurrent operation of multiple sonar sensors could cause cross-sensor acoustic interference problems. The contamination caused by such sensor crosstalk severely degrades the performance and sometimes, more seriously, mangles the whole sensor system. Basically, we have to mediate a sensor configuration including working frequencies of all sensors and their mounting positions to avoid sensor crosstalk. For mission sensors, we can selectively turn on the sensors as needed, considering other sensors installed on the mother ship. Additionally, potential problems of bio fouling and corrosion of underwater sensors should be taken into consideration and the quality of measurements should be regularly maintained.

Health-monitoring sensors are also very important. Even though they are small and relatively cheap compared to navigation sensors, it could save a robot. Health-monitoring sensors cover multipoint temperature and humidity values inside pressure vessels, status of the power system, for example, status of a battery using a battery management system (BMS) or status of tether cable using a ground fault-monitoring system, and status of the control system. At the same time, an emergency plan should be prepared for cases in which the system is headed toward complete failure. Examples include a situational warning for high temperature or clear thruster commands when communication is lost. An emergency beacon and a flash have to be operated independently from any system of the robot.

25.5 Computers, Communications, and Architecture

Design of computer systems, software architecture, and communication systems is a major task in designing an underwater robot. Designing the overall hardware and software structures and selecting appropriate components are very challenging and crucial. Nowadays, tethered robots or ROVs use a fiber-optic cable for real-time data transfer between a simple on-board computer system and a high-performance computing system on a mother ship. Untethered robots or AUVs require high computing power resources on-board with an underwater wireless communication system such as an acoustic modem.

It is worthwhile to note the Woods Hole Oceanographic Institute (WHOI)'s hybrid ROV, also known as *Nereus* [25.8]. It operates in two different modes. For broad area survey, the vehicle can operate untethered as an AUV capable of exploring and mapping the sea floor with sonars and cameras. For close-up imaging and sampling, *Nereus* can be converted at sea to operate as an ROV, using a lightweight, microthin, fiber-optic tether. *Nereus* uses its manipulator by teleoperation via the fiber-optic tether while the vehicle can be considered an AUV. In May 2009, *Nereus* explored the Mariana Trench, being the first vehicle to explore the Mariana Trench since 1998.

25.5.1 On-Board Computers

ROVs rely on computing systems on a mother ship and typical candidates are industrial PCs installed on a rack or on a shelf inside a control van as a surface control system. The overall system consisting of several PCs with a high-speed network handles data from the ROV and commands from its operator in order to control the ROV. Each computer is designated for a certain mission such as control, navigation, etc. However, the recent trend is to simplify the interface by an integration of multiple PCs and by specific software with expectation of cost saving benefits [25.39].

Unlike ROVs, AUVs must have compact, robust, reliable, and high-performance on-board computer sys-

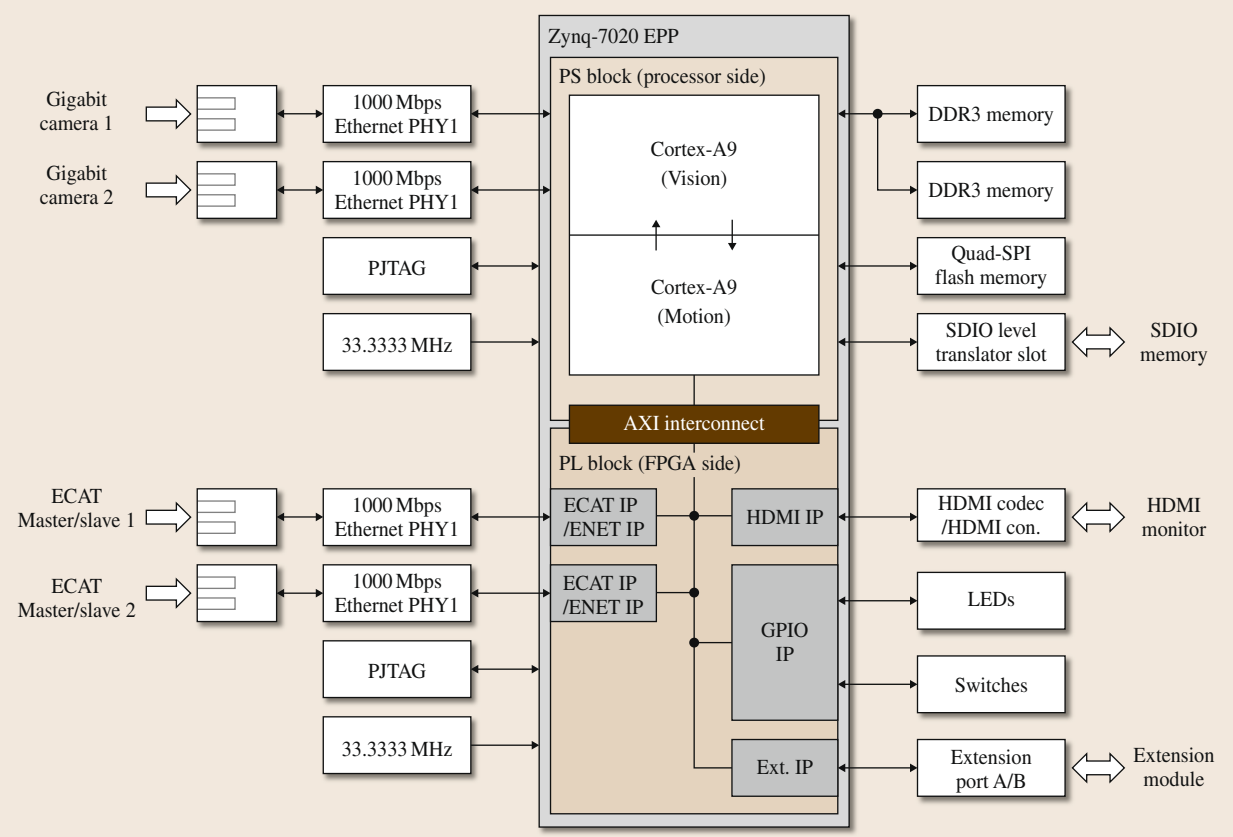


Fig. 25.3 Hardware configuration of Zynq-7020-based system, mController

tems. One popular choice is a VMEbus (Versa Module Europa) system [25.40]. Many AUVs before the year 2000 such as ODIN, Oceans Voyager II, and Aqua Explorer 1000 used a VME system based on Motorola CPU with a real-time operating system, such as VxWorks and OS9 [25.2]. Some AUVs like Kambara and Oberon use Compact peripheral component interconnect (PCI) systems [25.32, 41]. The controller area network (CAN) bus was originally designed as a vehicle bus standard to allow many subsystems such as actuators and sensors to communicate with each other using only two wires by a maximum 1 MHz data transmission rate [25.12]. The CAN bus could bring great advantages in designing the layout of wire harnesses for internal sensors of an AUV, but unfortunately, it has not been popular because there are few internal sensors that support it.

PC-104+ is an embedded computer standard which has no backplane but the boards are stacked on top of each other via stackable ISA, PCI, and PCIe bus connectors [25.33]. PC-104+ based systems are widely used in various types of embedded applications because of their advantages such as small size, stackable, rugged, compact, interoperable, PC compatible high-quality, low cost, and low power consumption [25.33, 40]. Basically, it consists of a CPU board with an optional power supply module. Other required functions can be added by stacking multiple peripheral modules such as frame-grabber, analog digital conveter/digital analog converter (ADC/DAC) converter, digital IO, relay, serial communication, CAN network, and GPS receiver from different manufacturers. Each module offers high-performance specifications and environmental specifications, such as extended operating temperature from −40 to +85 °C. As an example, a decade ago, ODIN, originally developed with a VME-based onboard computer system, was refurbished to a PC-104+ based AUV, ODIN III [25.42]. Today, many recently developed AUVs employ PC-104+ based systems. It is also noted that as a possibility of a heterogeneous bus system, there is a PC-104+ bus-based system (CPU module) along with standard VME-based I/O modules using a protocol interface card [25.40].

PC-104+ is a good choice for a single CPU system. However, when we need more computing power for implementing complicated algorithms, additional PC-104s must be installed because multiple CPUs

Avidbots Ex1016
Page 646 of 2154

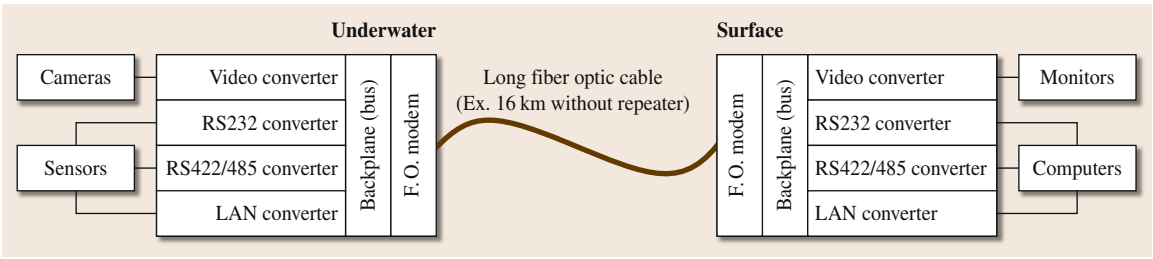


Fig. 25.4 Typical diagram of fiber-optic communication system for ROV

are not supported in a PC-104+ stack. For example, yShark [25.43] has three PC104+s with a high-speed local area network (LAN) for robot control, image processing, and acoustic signal processing. This causes many difficulties, such as a complicated system structure, wiring, space, and cooling in a pressure vessel. Alternative design of a distributed on-board system for an AUV is a combination of a PC-104+ system and an ARM-based system. Besides consumer mobile devices, many mobile robots have used ARM processors 32-bit RISC (reduced instruction set computer) microprocessors because of its high-performance, low power consumption, and low cost. As an example, Fig. 25.3 shows a hardware configuration of mController (announced by Redone technology, Korea) which is based on the Xilinx Zynq-7020 platform [25.44]. Its size is only 125 mm × 125 mm, and it consists of Dual core Cortex-A9 for typical motion control and vision processing, and an FPGA part for data parsing of various sensors. In this hardware hierarchy, PC-104 could be designated to operate high-level tasks.

An on-board-independent microcomputer-based emergency response system is very important to carry out critical functions, such as power disconnection of thrusters and weight drop when the on-board main system is dead. The emergency response system consists of a microprocessor that monitors the status of the on-board main system, an actuator, and a backup battery. Under normal conditions, the on-board main system would regularly check the status of the emergency response system.

25.5.2 Communications

For ROVs, an umbilical cable is used for communication between an ROV and its mother ship. An umbilical cable commonly has a coaxial core or a fiber-optic core. A fiber-optic cable is much better than a coaxial cable in terms of performance and physical dimensions. Light and thin fiber-optic cables offer high-speed, secured, long-distance data transmission without electromagnetic interference, and less drag force on an ROV, even though it is an expensive solution mainly because of optic fiber distribution devices, fiber-optic tools, and the special knowledge required for a handling fiber core. An umbilical cable of a typical ROV has been designed to have 2–3 fiber-optic cores including a spare core. A fiber-optic modem connected to each end of a fiber-optic cable sends and receives data at the same time, working together with several sub-boards that convert data in various formats, for examples, NTSC Video, RS232, and LAN as shown in Fig. 25.4.

Since the demand of high-quality video, such as multiple HD video, 3-D HD video, and UHD (ultrahigh definition) video is rapidly increasing, 100 Gbps optical communication systems are released to handle mass data using a technology known as dense wave division multiplex (DWDM) [25.12]. Cable designers are also making an effort in developing new techniques to add more fiber-optic cores inside an umbilical cable, where each core could be designated to high-quality video devices. The *Nereus* team used a light fiber tether system to send a lightweight battery-powered vehicle to the full depth of the ocean of about 11 km in the ROV mode. The selected Sanmina/SCI buffered fiber is 0.25 mm in diameter, weighs only 0.173 kg for 11 km cable in water, has a working strength of 8 N, and a breaking strength of 108 N [25.5].

For untethered vehicles like AUVs, wireless communication is considered. Typical radio communication would be fine when an AUV is on the surface. In an underwater environment, acoustic communication is the practical solution at the moment even though a commercial acoustic modem has a limited range and a very limited data rate. Table 25.6 shows the specifications of commercial acoustic modems. Acoustic communication is relatively slow compared to radio communication, and it is fundamentally limited by the speed of sound in water, roughly 1500 m/s. Because acoustic attenuation increases with sound frequency, lower frequencies are desirable for long-range communications. But, to generate signals at very low frequencies, larger size transducers are needed. This results in lower data rates that are undesirable [25.2, 13]. The speed and direction of sound propagation vary depending on various factors such as surface waves, currents, sea state, tides,

Table 25.6 Specifications of commercial acoustic modems

Specification	A modem	B modem	C modem	D modem
Size (length × dia)	585 × 183	200 × 165	235 × 126(87)	580 × 150
Material	Aluminum	Aluminum		
Weight (air/water)	11/6	5/1.5	4.2/2.3	21/(n/a)
Voltage	18–50 V_{DC}	10–20 V_{DC}	12–24 V_{DC}	18–28V_{DC}
Power (transmit/receive/sleep)	< 50 W/1/0.03	20/0.6/0.005	2/0.75/0.008	40/0.9/0.009
Working depth	3000	1000	200	7000
Working temperature (°C)	−5–40	−5–40	−5–45	−5–45
Beam width	±25	Hemispherical	210 (Omni)/120 (wide)/70 (narrow)	210 (Omni)/70 (Directional)
Working range	7000 m	Up to 20 km	350 mm	7000 m (Omni)/10 000 m(D)
Frequency	14–22 kHz	7.5–12 kHz	26.77–44.62 kHz	7.5–12.5 kHz
Modulation	QPSK	MFSK and DPSK	Hybrid	Hybrid
Raw data rate	1.5–15 kbps	2 kbps	9.6 kbps	2.5 kbps
User data rate	0.6–10 kbps	0.3 kbps	7 kbps	2.0 kbps
Bit error rate	10^{-9}	10^{-6}	10^{-9}	10^{-7}
Environment			Near vertical, horizontal	Near vertical, horizontal

DPSK: differential phase shift keying

temperature, conductivity, depth, and sea-bottom type. Hence, acoustic communication is inherently delayed, due to propagation delay. Several techniques in signal processing, data packaging, and coding schemes have been developed to overcome these difficulties due to signal absorption, geometric spreading losses, boundary effects, and multipath through.

Acoustic modems consist of three main parts: an underwater transducer, an analog transceiver including a matching preamp and an amplifier, and a hardware platform for control and signal processing, such as microprocessor, digital signal processor (DSP), or field programmable gate array (FPGA) [25.45]. When acoustic communication technology was in its infancy, an analog-based hardware platform was developed. Then digital techniques brought important improvements for increasing the reliability of transmissions using explicit error-correction techniques and increasing some level of compensation for the channel reverberation both in time (multipath) and frequency (Doppler spreading) [25.46]. There are several methods in digital modulation, where the most fundamental techniques are PSK (phase shift keying), FSK (frequency shift keying), ASK (amplitude shift keying), and quadrature amplitude modulation (QAM), and their improved versions are MFSK (multiple FSK), QPSK (quadrature phase shift keying), MPSK (M-ary phase shift keying), MSK (minimum shift keying), GMSK (Gaussian minimum shift keying), MQAM (M-ary quadrature amplitude modulation), and so on.

Just like a typical network, a data packet is packed with a header and a tail for addressing, error detection and correction not to be corrupted by transmission errors. In particular, the data link layer protocol implemented in some modems detects errors in a received packet and then requests retransmissions, which degrades the effective transfer rate. To reduce the retransmission rate, some modems have a function to adjust the speed of transmission depending on the channel condition. Another important technique is about minimizing the effects of multipath, which is most prominent over long ranges and shallow water because the original signal can be bounced between the surface and bottom before arriving at the receiver. There are a few algorithms for robust data transmission in multipath environments, such as convolutional coding, multipath guard period, and data redundancy. Also, a time-reversal technique is an alternative to reducing the effects of multipath [25.12]. From a practical point of view, acoustic communication has some other limitations, such as horizontal communication in shallow water, the loss of line of sight between the modems, the in-band noise (typically caused by another sonar system), echo sounder from ships, a shallow zone, and clouds of air bubbles near a receiver. These limitations might degrade its performance or make it unreliable.

Electromagnetic waves are strongly absorbed by sea water, but, for close-range communication, electromagnetic communication would be useful because it would be less affected by multipass interferences. JAMSTEC (Japan Agency for Marine-Earth Science and Technology) has developed a new communication using electromagnetic waves. This was experimentally evaluated up to 50 m distance [25.12].

Another approach is an optical underwater communication that achieves much higher data transfer rates

than an acoustic communication system at significantly lower power consumption and smaller packaging. However, the operation requires a point-to-point communication setting, requiring the alignment of the receiver and the transmitter for the communication to work effectively. Their range and scope are also affected by the water clarity, water light absorption, and power loss due to propagation spherical spreading [25.47]. Recently, the Massachusetts Institute of Technology (MIT) proposed a real-time video delivery solution based on underwater optical communication, AquaOptical II that consists of a high-bandwidth wireless optical communication device and a two-layer digital encoding scheme designed for error-resistant communication of high-resolution images. Its maximum transmission bandwidth of 4 Mbps and various video streaming tests showed similar performances at close ranges of up to 25 m in a pool trial [25.48], and up to 3–4 m on the Charles River near Boston. It requires accurate motion stabilization because of the narrow beams involved [25.49].

When an AUV is on the surface or an AUV exposes an antenna above the surface, radio communication is the best option. For a short range in which an AUV is operating near shore or its support craft, 900 MHz wireless modems or 2.4 GHz 802.11 wireless networks are common considering antenna sizes [25.49]. For longer ranges, radio links make use of an earth orbiter satellite like Argos satellites and Iridium satellites. In some cases, we have to make our own pressure-tight antenna, including a motion stabilizer to track the satellite regardless of fluctuation of motion. A tracking system for the *Urashima* vehicle was developed using data on network design and the oscillating characteristics of the vehicle [25.12]. Oscillation of the vehicle was measured with the high-accuracy inertial navigation system installed in *Urashima* in a sea trial. An oscillation angle of 7° and period of 0.15 Hz were estimated.

25.5.3 Software Architecture

includes various software components as shown in Fig. 25.5. The lower part of the software architecture is middleware as a class of software technologies designed to help managing the complexity and heterogeneity inherent in distributed systems [25.50]. It is defined as a layer of software above an operating system but below application programs that provide a common programming abstraction across a distributed system.

In the literature, AUV architectures were classified into four categories – the hierarchical architecture, the hierarchical architecture, the subsumption architecture, and the hybrid architecture [25.51] or into three categories – the deliberative architecture, the reactive architecture, and the hybrid architecture [25.52]. Since most of the recently developed architectures are hybrid, instead of describing each architecture, key features of recently proposed software architectures for AUVs are described below.

Teleo-Reactive EXecutive (T-REX) was developed for a specific underwater robot by the Monterey Bay Aquarium Research Institute [25.53–55]. It is a goal-oriented hybrid executive with an embedded automated planning adaptive execution using agents. In order to make the embedded automated planning scalable, the T-REX enables the scope of deliberation to be partitioned functionally and temporally inside units called Teleo-reactors. A Teleo-reactor agent is considered the coordinator of a set of concurrent control loops, which is characterized by functional scope, temporal scope, and timing requirements.

Huxley was developed by Bluefin Robotics originally for its fleet of AUVs. Since it is for their wide range of commercial products, the most fundamental requirement was the flexibility in terms of robustness and reliability, extensibility for new hardware, maintainability, and testability [25.56]. It consists of two layers – a reactive layer and an executive layer – and it allows expanding these core layers through a standard interface. Another key element is the Huxley messaging protocol, called the stream-oriented messaging architecture (SOMA), where SOMA is a publish-subscribe messaging protocol. Huxley's layered architecture with SOMA and SOMA interfaces can be extended along with new functionalities and/or modified configurations of future AUVs while localizing changes and increasing the reusability of components.

The lightweight communications and marshalling (LCM) system was designed as an alternative for implementing inter-process communication (IPC) for real-time robotics in the marine environment. It was

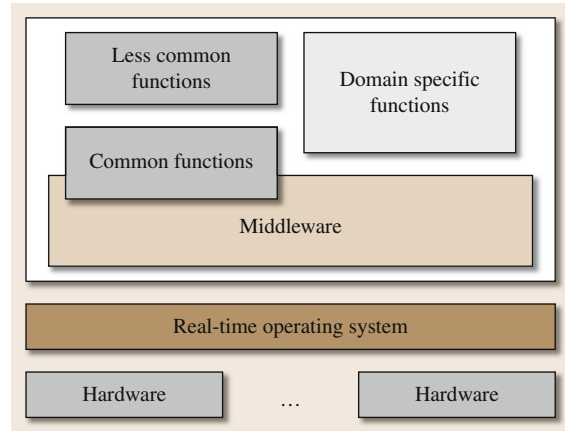


Fig. 25.5 Typical configuration of software for robot

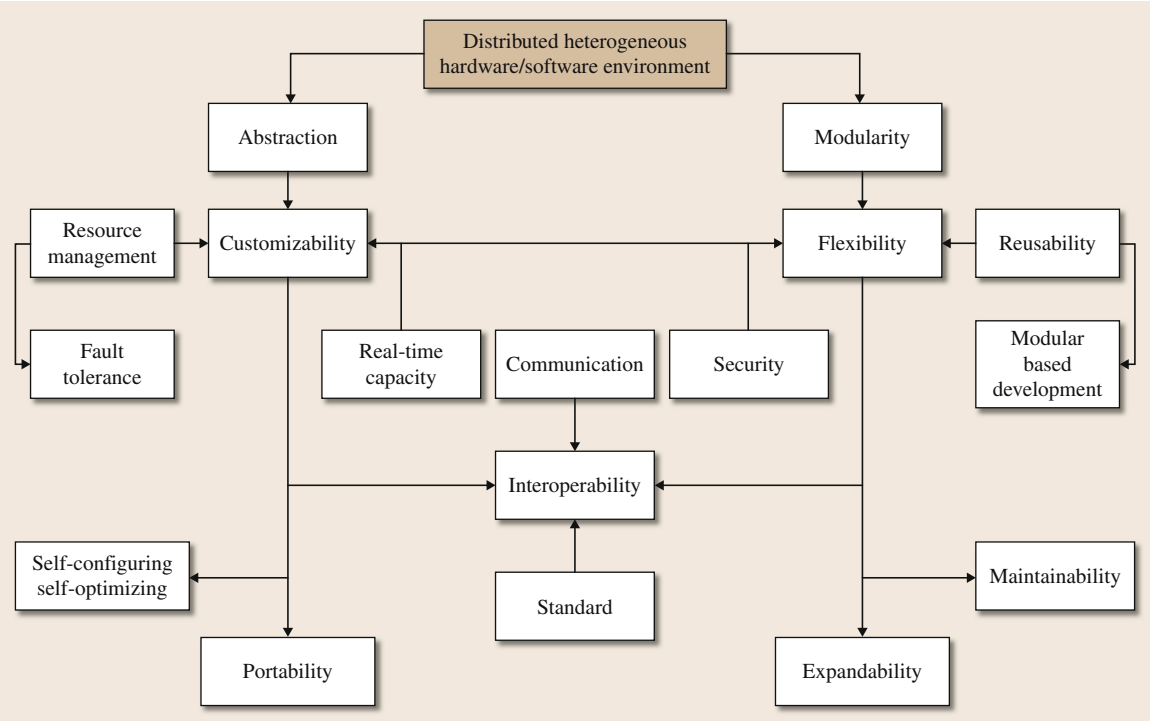


Fig. 25.6 Key characteristics of middleware

developed specifically for low-latency, high-throughput communication accomplished by a set of software tools for message passing between modules using user datagram protocol (UDP) multicast, marshalling (encoding and decoding) of LCM messages, language-independent data structures for messages of any digital data, allowing the transparent distribution of the system. This architecture was verified by two marine robotics applications [25.57].

MOOS-IvP consists of two distinct open source software projects. The Mission Oriented Operating Suite (MOOS) is, as a core middleware, a suite of libraries and executables developed by the Mobile Robotics Group at the University of Oxford. It provides inter-process communication using a publish-subscribe model, as a star topology where all messages go through a central MOOS server. IvP (interval programming) refers to a multiobjective optimization method used by the IvP helm for arbitrating between competing behaviors in its behavior-based architecture. MOOS provides modules for navigation, control and data logging, tools for mission replay from log files and communication debugging, and also provides driver modules for some sensors widely used with AUVs [25.57–59]. MOOS-IvP has been used for several AUVs including the Bluefin 21 in. UUV, the Hydroid REMUS-100 and REMUS-600 UUVs, the Ocean Server Iver2 UUV, the Ocean Explorer 21 in. UUV, autonomous kayaks from Robotic Marine Systems and SARA Inc., and two larger USVs from the NATO Underwater Research Center in La Spezia Italy [25.54, 58].

Let us look at middleware that should be easy to use, robust against change, reliable for any faults, easy to maintain, efficiently flexible, and supportive for heterogeneous and distributed hardware components. Well-structured proven middleware with some often-needed and value-added functions would minimize time and cost for development. In the literature [25.50–53, 56–58, 60–64], there are many demanded characteristics to realize above expectations, as shown in Fig. 25.6. Fundamental characteristics of middleware are abstraction and modularity. Abstraction is expected to simplify and to lessen burdens of application development process, and modularity is a necessary condition of flexibility and customizability. Abstraction allows developers to concentrate on building advanced algorithms themselves by hiding details of low-level hardware resources of heterogeneous and distributed hardware environments [25.51, 62]. This concept includes more specified subissues: interoperability by using simplified interfaces with communication and standard; customizability with real-time capacity; and flexibility and resource management. Additionally, resource management handles faults in hardware. From these characteristics,

a middleware is free from the specific hardware environments, and could obtain portability, self-configuring and self-optimizing. To achieve flexibility, we have to follow a modular-based development for increasing reusability of existing components based on modularity and interoperability. This development approach allows us to focus on building modules without considering specific relationships between existing modules. Based on this, expandability and maintainability can be realized, where expandability is the ability to incorporate new resources, for example, newly developed sensors without interfering with the existing parts, and maintainability is about how easy to embrace modification of components without affecting the rest. To implement a middleware, standards such as the common object request broker architecture (CORBA), the Internet communications engine (ICE), and the open dynamics engine (ODE) are applied [25.64], which bring many advantages. Basically, it requires less effort in implementing and evaluating a core engine from scratch, and some of these have additional functionalities such as standardized interface and security. Security aspects such as authentication, authorization, and secure communication become essential for collaborations of multiple robots or classified applications [25.64]. So far, many research groups have proposed their own middleware for their robots, such as MIRO (middleware for robot), ORCA (open robot control architecture), UPnP (universal plug and play), RT (robot-technology), the player and stage, OPRoS (the open platform for robotic services), ROS (robot operating system), OROCOS (open robot control software), ERSP, MRDS (Microsoft robotics developers studio), MOOS (the mission oriented operating suite), and Webots [25.50].

A development of middleware itself is not a trivial work. Understanding concepts is one thing and implementing them is another. The reasons for possible errors are mainly:

1. Many inherently distributed and heterogeneous components which will vary to improve performance.
2. The limited and restricted resources available on a robot platform.
3. The high rate of system failures from incorporating complicated algorithms and interacting with the environment, causing exceptional situations [25.50, 62].

Basically, an underwater robot's middleware has almost the same requirements as a robot, but the weighting of requirements and the point of view of requirements are little different. For an underwater robot's middleware, the following points must be considered:

1. *Limitation of resources*: Underwater robots use a low bandwidth of an underwater communication and thus, has to manage a mission using internal computing resources.
2. *Unstable sensor information*: Underwater sensors based on sonar are not stable and accurate enough to directly use data for robot control systems. Therefore, assisting algorithms such as well-tuned filtering algorithms and sensor fusion algorithms are needed.
3. *Lack of accessibility*: Underwater robots do not allow any access of an electrical contact used for debugging because of its water-proof pressure vessel.

This is one of the most challenging aspects in building an underwater robot. Thus, for the underwater robot, middleware should provide an independent channel for a supporting a debugging process.

No middleware is perfect, but we believe that we can achieve significant improvements through trial and error coupled with extensive feedback from the community [25.50]. When you develop system software and its architecture on your own, the implementation of abstraction and modularity can be understood with the white-box framework and the black-box framework concepts. The black-box framework supports extensibility by defining interfaces for components without detail information inside the components. White-box, as a base class in a class library can be designed for expanding with subclasses, based on object-oriented language features [25.63]. In the early stage of development, software architecture is mainly conceived as a white box framework. Hence, information inside the boxes, that is, classes, should be understood. As the architecture becomes more advanced and the components become more concrete, the black-box framework is the solution for most common expansions. At the moment, MOOS-IvP would be the best choice to start developing your vehicle software because it is open source software, and there is a relative wealth of information and case examples for AUVs online.

As an alternative to formal software architectures discussed above, we propose a simple but effective solution in testing a robot as shown in Fig. 25.7. This architecture has one real-time thread with several non-real-time threads for controlling actuators and devices installed on the robot.

In the Fig. 25.7:

1. The packet conveys three types of commands from an operator – commands for teleoperation, commands for RT-thread (real-time thread) itself and

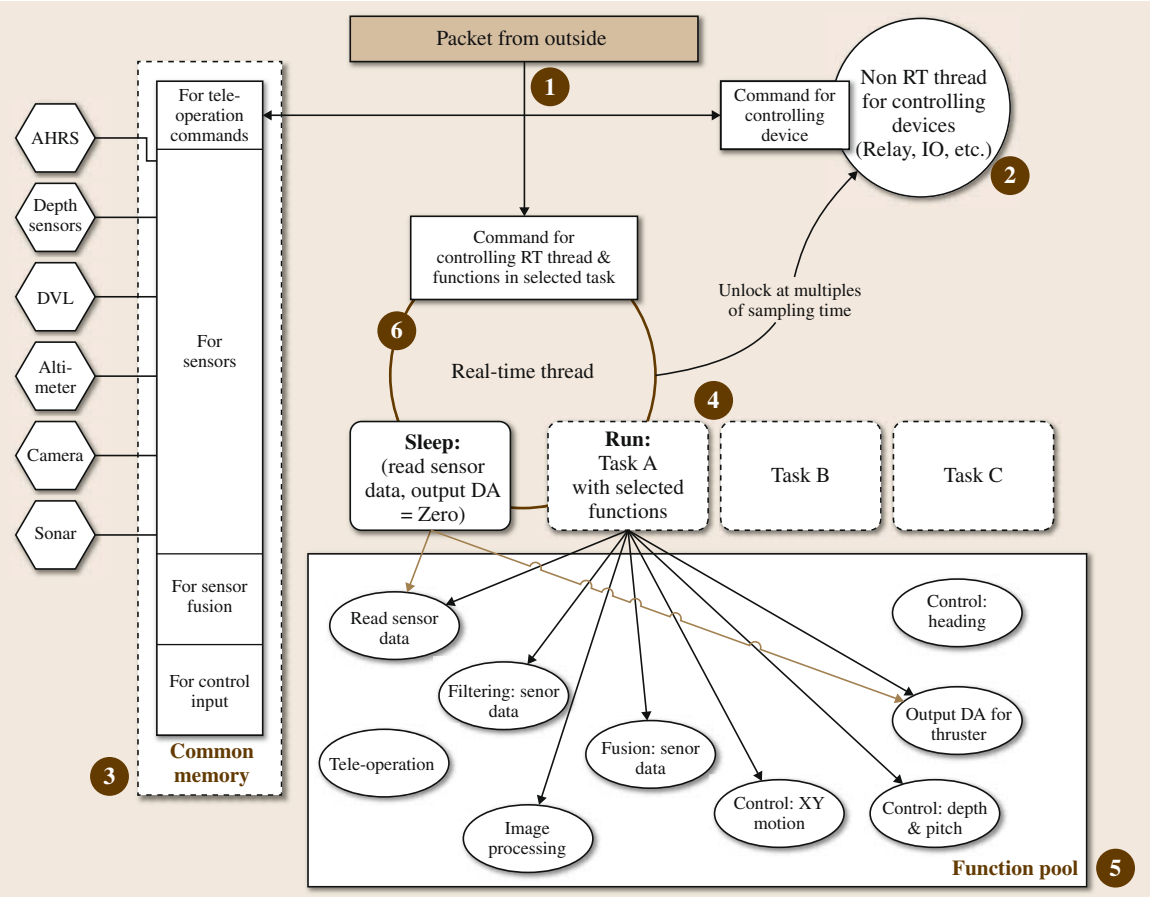


Fig. 25.7 Example diagram of simple software architecture

functions in the selected tasks, and commands for devices of the robot.

2. Device control commands run in a non-RT thread triggered by the RT-thread, typically every 1 s.
3. All sensor data including teleoperation commands and results of any sensor fusion algorithms are independently saved in common memory, and thruster commands, that is, control input are also saved here.
4. This architecture has a manual task coordinator and each task can be easily synthesized using functions from a function pool.
5. A robot is basically in one of two modes, sleep and run, where the sleep mode is doing only health-monitoring with basic sensors, the run mode runs a selected task which consists of subtasks in function pool. Tasks are designed for specific purposes, and one of them is selected by an operator before running. Of course, the default task should be the teleoperation task.
6. The RT-thread carries three main functions: doing any critical device control commands first, DA converting for control input of previous sampling time saved in the common memory, and processing algorithms using a sensor input saved in the common memory.

Since the function of this RT-thread is based on a basic sampling time, we could easily implement any multisampling time structures for various latency times of sensors and load-distributing structures to enhance the usage of computing resources. This would be done by calling functions at multiples of the basic sampling time. This architecture has been applied to several test-bed AUVs and smart ROVs [25.42, 43, 65] (VIDEO 797, VIDEO 799).

Avidbots Ex1016
Page 652 of 2154

25.6 Underwater Manipulators

Most commercial underwater robots with manipulators are ROVs. A typical work-class ROV for intervention missions has a 7 degrees-of-freedom (DOF) manipulator and a 5 DOF grabber. The manipulator is used for interactive manipulation by a human operator on the mother ship, while the grabber is used to maintain a relative position of ROV to the target object by grabbing a target structure in the presence of an underwater current or wave. This section describes underwater manipulator systems based on materials presented in [25.66]. Table 25.7 lists samples of heavy duty underwater manipulators currently available in the market. They are quite expensive but utilize proven technologies in the field. However, most commercial underwater manipulators are hydraulic-driven and developed for ROVs and they are not suitable for AUVs that have limited on-board power and whose on-board sensors could be sensitive to noise. Therefore, electric-driven manipulators would be preferred for AUVs like SAUVIM.

A typical work-class ROV for intervention missions requires two individual operators, one steering the vehicle and the other operating the manipulator. One of the main tasks of the vehicle operator is to keep the ROV as steady as possible (station-keeping) since the vehicle motion can disturb the manipulator and greatly affects the accuracy of its end-effector interacting with the environment. The accuracy of the end-effector would be greatly improved if the manipulator controller could compensate for any disturbance due to the motion of the vehicle. However, most commercial ROV systems do not use vehicle motion compensation control for the manipulator. In the past, a very few AUV systems for intervention missions have been reported and they were used as a test-bed for developing autonomous manipulation or as a working model for research.

25.6.1 Dynamics

The analysis of the dynamics of underwater vehicle manipulators is much more complex than that of regular manipulators on the factory floor. For example, it is difficult to accurately introduce the effects of the added mass and the added moment of inertia in the modeling process for multiple links of the manipulator. The frictional force and the drag force due to the velocity of the manipulator itself, waves, and currents are also complex in modeling. The modeling and control of underwater manipulators were studied by several researchers, including [25.67–78]. In this section, it is not our intention to derive details of the full dynamic equations of the underwater vehicle manipulators. Instead, we would like to point out several specific elements that differentiate underwater vehicle manipulators from regular manipulators. The hydrodynamic forces acting on a rigid body moving in fluid can be found in [25.79, 80]. Similar to the vehicle main body, each link of the manipulator will be influenced by hydrodynamic forces including added mass, buoyancy acting at the center of buoyancy of the link, the fluid acceleration force resulting from the acceleration of the fluid itself, and the fluid drag forces exerted on the link.

As the profile drag forces are dominant for slowly moving objects, the drag force exerting on an infinitesimal element of the i-th link and the total drag force on the link can be expressed as

$$\mathrm{d}D_{\mathrm{f}i} = \frac{1}{2}\rho C_{\mathrm{D}} b_i v_{\mathrm{r}i} |v_{\mathrm{r}i}| \,\mathrm{d}x \,, \tag{25.1}$$

$$D_{\mathrm{f}i} = \frac{1}{2}\rho \int_0^l C_{\mathrm{D}} b_i v_{\mathrm{r}i} |v_{\mathrm{r}i}| \,\mathrm{d}x \,, \tag{25.2}$$

where C_{D} is the drag coefficient, ρ is the fluid density, $v_{\mathrm{r}i}$ is the relative velocity of the element to the fluid flow velocity, $b_i \,\mathrm{d}x$ is a projected area of the element, and l is the link length.

Considering the effects of the hydrodynamic variables above and assuming that the vehicle is stationary during the manipulation, the dynamic model of an underwater manipulator having a series of links and joints can be represented by the following equations,

$$\begin{aligned}&\mathbf{M}_{\mathrm{m}}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}_{\mathrm{m}}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \boldsymbol{D}_{\mathrm{m}}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \boldsymbol{F}_{\mathrm{m}}(\dot{\boldsymbol{q}}) \\ &\quad + \boldsymbol{G}_{\mathrm{m}}(\boldsymbol{q}) + \boldsymbol{\tau}_{\mathrm{d}} = \boldsymbol{\tau}_{\mathrm{m}} \,,\end{aligned} \tag{25.3}$$

where $\boldsymbol{q} \in \boldsymbol{R}^{\mathrm{n}\times 1}$ is a joint angle vector, $\boldsymbol{\tau}_{\mathrm{m}} \in \boldsymbol{R}^{n\times 1}$ a joint torque vector, $\mathbf{M}_{\mathrm{m}}(\boldsymbol{q}) \in \boldsymbol{R}^{n\times n}$ an inertia matrix including added mass terms, $\boldsymbol{C}_{\mathrm{m}}(\boldsymbol{q}, \dot{\boldsymbol{q}}) \in \boldsymbol{R}^{\mathrm{n}\times 1}$ a nonlinear vector arising from centrifugal and Coriolis effects including added mass terms, $\boldsymbol{D}_{\mathrm{m}}(\boldsymbol{q}, \dot{\boldsymbol{q}}) \in \boldsymbol{R}^{n\times 1}$ a nonlinear vector due to hydrodynamic forces, such as drag forces, $\boldsymbol{F}_{\mathrm{m}}(\dot{\boldsymbol{q}}) \in \boldsymbol{R}^{n\times 1}$ a nonlinear vector due to friction at the manipulator joints, $\boldsymbol{G}_{\mathrm{m}}(\boldsymbol{q}) \in \boldsymbol{R}^{n\times 1}$ a nonlinear vector due to gravity and buoyancy forces, $\boldsymbol{\tau}_{\mathrm{d}} \in \boldsymbol{R}^{n\times 1}$ a vector of unknown signals due to unmodeled dynamics or external disturbances such as current.

During the manipulation, the end-effector or tool of the manipulator can be in contact with the environment, and forces and moments would occur at the end-effector. In this situation, (25.3) must be modified by adding the reaction torque to the end-effector force on the right-hand side of (25.3)

$$\boldsymbol{\tau}_{\mathrm{r}} = \mathbf{J}_{\mathrm{m}}^{\mathrm{T}}(\boldsymbol{q}) \boldsymbol{f}_{\mathrm{e}} \,, \tag{25.4}$$

Table 25.7 Samples of commercially available underwater manipulators

Specification	Eca robotics	International submarine engineering (ISE)	Kraft telerobotics	FMC/Schilling robotics	Western space and marine
Model	Arm 7H	Magnum-7	Predator	Titan 4	The ARM
DOF	6 plus gripper	6 plus gripper	6 plus gripper	6 plus gripper	6 plus gripper
Power source	Hydraulic	Hydraulic: max 1000 psi/19 lpm (5 gpm)	Hydraulic: 103 bar–207 bar (1500 psi–3000 psi), 19 lpm (5 gpm)	Hydraulic: 103 bar–207 bar (1500 psi–3000 psi), 19 lpm (5 gpm)	Hydraulic: max 204 bar (3000 psi), 7.6 lpm (2 gpm)
Material	Titanium or aluminum	Aluminum with stainless steel fittings	Anodized aluminum and stainless steel	Titanium	
Max grip force	490 N (110 lbf)	2009 N (450 lbf)	1334 N (300 lbf)	4092 N (920 lbf)	1467 N (330 lbf)
Wrist torque	108 Nm (80 ft – lbs)	108 Nm (80 ft – lbs)	135 Nm (100 ft – lbs)	170 Nm (125 ft – lbs)	
Actuator	Hydraulic cylinder	Hydraulic cylinder	Hydraulic cylinder	Hydraulic cylinder	Hydraulic cylinder
Max reach	1700 mm (66″)	1524 mm (60″)	2019 mm (79.50″)	1920 mm (75.7″)	1700 mm (66″) from azimuth axis to fingers
Max lift capacity		454 kg (1000 lbs)	227 kg (500 lbs)	454 kg (1000 lbs)	45.4 Kg (100 lbs)
Max lift at full extent	90 kg (198 lbs)	295 kg (650 lbs)	91 kg (200 lbs)	122 kg (270 lbs)	29.5 kg (65 lbs)
Working depth (standard)	7500 msw (24 600 fsw)	6000 msw (19 700 fsw)	3000 msw (9800 fsw)	4000 msw (13 100 fsw)	Unlimited
Working depth (Extended)	Unlimited	11 000 msw (36 000 fsw)	6500 msw (21 000 fsw)	7000 msw (22 967 fsw)	Unlimited
Homepage	www.ecarobotics.com	www.ise.bc.ca	www.krafttelerobotics.com/	www.fmctechnologies.com/Schillingrobotics.aspx	www.wsminc.com

where $\mathbf{J}_m \in \boldsymbol{R}^{6\times n}$ is the manipulator Jacobian matrix, and $\boldsymbol{f}_e \in \boldsymbol{R}^{6\times 1}$ a vector of forces, and torques at the end-effector.

The dynamics of the underwater vehicle manipulator is quite complicated, highly nonlinear, and involves coupled equations with unknown parameters and disturbances as shown in (25.3). It is almost impossible to accurately model the dynamic equations and to operate the manipulator at a nominal speed using a conventional controller. However, when the manipulator moves very slowly in a friendly environment with no or minimal current or wave, the dynamics of the manipulator can be represented by the simplified linear version of (25.3) since the effect of velocity-dependent terms become negligible. In fact, during the actual operation in the field, most commercial underwater manipulators move very slowly (much less than 1 rad/s) for safety and other reasons, and they use conventional joint controllers for the master-slave teleoperation.

25.6.2 Teleoperation

A typical setup for teleoperation involves the operator sitting in the control room located on the mother ship and holding a master arm that is a miniature of the actual underwater vehicle manipulator (slave arm) attached to the ROV. The operator controls the slave arm by moving the master arm and its motion is followed by the slave arm. During the operation, the operator relies on visual information of the work site in the form of a series of two-dimensional (2-D) video images that are captured by cameras on the vehicle main body and the slave arm. The images are then transmitted to the control room monitors by an underwater cable. The view of the work site is not only limited, but the visual information especially for the deep sea operation is also often delayed. If the vehicle is not holding or sitting on an underwater structure, the ROV operator must try to control the vehicle for station-keeping during the manipulation. However, even if there is no current or wave during the station-keeping, the vehicle moves like free floating within the accuracy of the vehicle position sensors in the order of a meter or so. Therefore, it is difficult to achieve the accuracy at the end-effector since underwater vehicle manipulators are attached to the vehicles that are constantly moving, unlike industrial manipulators, whose bases are fixed on the factory floor. As mentioned in [25.81], many simple tasks on the ground such as plugging become very difficult to perform by the underwater vehicle manipulator. Underwater plugging tasks, done on a trial-error basis as the required precision cannot be achieved by the ROV manipulator

system, take hours to complete. Therefore, operator fatigue often becomes a critical issue in tele-operating the underwater vehicle manipulator.

25.6.3 Autonomous Manipulation

Autonomous manipulation on a moving base, such as terrestrial mobile robots, humanoids, and underwater robotic vehicles is a very challenging task in the area of robotics in general, especially in unstructured environments, such as underwater. It is defined as the capability of a robot system that performs intervention tasks requiring physical contact with unstructured environments without continuous human supervision. Unlike industrial manipulators which have fixed bases on the floor, autonomous manipulation requires a system capable of assessing a situation, including self-calibration based on sensory information, and executing or revising a course of manipulating action without continuous human intervention. Therefore, developing a system capable of fully autonomous manipulation would be a great achievement and make a substantial impact on a variety of application areas with significant economical, societal, and scientific importance [25.82].

Let us look at one scenario of cutting an underwater cable. With given information about the cable's location and shape, the vehicle has to navigate to the location, identify the cable, position itself, and cut the cable. It may sound like a very simple task. However, it would not be so simple if it has to be done in autonomous mode. Even in teleoperation mode, it would not be an easy task without the coordinated motion control if the vehicle is floating in water. In fact, the low bandwidth and significant time delay inherent in acoustic subsea communications represent a considerable obstacle to remotely operate a manipulation system, making it impossible for remote controllers to react to problems in a timely manner. Nevertheless, robots for autonomous underwater intervention would pave the way for a different range of new operations, such as deep-ocean and under-ice exploration, tasks in hazardous areas, tasks in natural or man-made disastrous regions, automated searches, and surveillance missions, to name a few.

In the past, many researchers have studied advanced control of AUV itself as reported in [25.83–91], but only a few AUVs with manipulators were introduced. OTTER is an AUV equipped with a single degree-of-freedom arm, which was designed to be used as a test-bed for developing autonomous technologies at



Fig. 25.8 The SAUVIM vehicle (Sand Island, Hawaii, 2008)

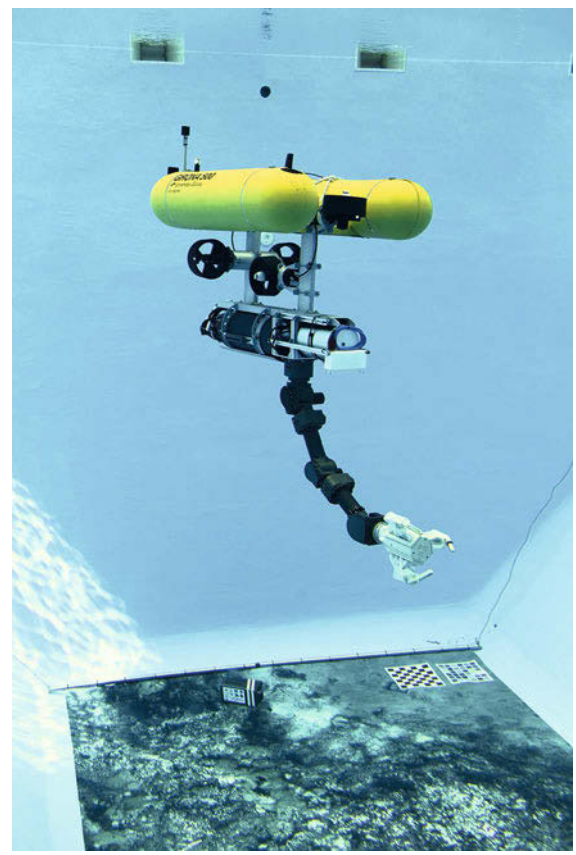


Fig. 25.9 Girona 500 I-AUV

the Stanford Aerospace Robotics Lab in 1996. It is a hovering-capable underwater vehicle that operates in a test tank at the Monterey Bay Aquarium Research Institute (MBARI) [25.92]. Another intervention AUV, namely ALIVE, was developed by Cybernetix in 2003. The aim of the EU-funded ALIVE project was to develop an intervention-AUV capable of docking to a subsea structure, which has not been specifically modified for AUV use. A description of the ALIVE vehicle was given in [25.93].

The key technology in underwater intervention performed with autonomous vehicles is autonomous manipulation. In the literature, no such system fully functional for autonomous manipulation was reported until recent developments with a semi-autonomous underwater vehicle for intervention missions, SAUVIM [25.94–97]. SAUVIM (Fig. 25.8) funded by the US Office of Naval Research was jointly developed by the Autonomous Systems Laboratory (ASL) of the University of Hawaii, Marine Autonomous Systems Engineering (MASE), Inc. in Hawaii, and Naval Undersea Warfare Center Division Newport (NUWC) in Rhode Island. The first fully autonomous manipulation in an unstructured ocean environment was demonstrated with SAUVIM at Snug Harbor, Honolulu, Hawaii on January 20, 2010. Following its success, more underwater vehicles having the capability of autonomous manipulation are expected to develop in the near future. Among some exemplar, underwater intervention tasks are object recovery/rescue, and maintenance/repairing of underwater facilities. Therefore, one may consider a task like black-box recovery, grasping an object of interest and placing it at a desired location for benchmarks in future development. One recent development after SAUVIM is the TRIDENT project [25.98–101] in Europe, which developed Girona 500 I-AUV (Fig. 25.9). They performed an autonomous underwater intervention task, with a black-box recovery in Port de Sóllers harbor (Mallorca) in October 2012 [25.14].

As subsea business in oil and gas industry is about to boom in the very near future, AUVs for intervention missions will receive more attention. While autonomous underwater manipulation is still an active research topic, recent development with SAUVIM and Girona 500 I-AUV would accelerate advancement in the field.

25.7 Conclusions and Further Reading

In this chapter, we described major subsystems of underwater robots – mechanical systems, power systems, actuators, sensors, computing systems, software architecture, communications, and manipulator systems, and also discussed critical design issues from the authors' practical experiences.

There are currently more than 1000 work-class ROVs. They are used for various applications, such as scientific research, military operations, and underwater constructions. About 57% of ROVs are used for offshore oil and gas industry. Most commercial ROVs have used old but proven technologies in the past 20 years. However, the ROV operation is very expensive as it requires a mother ship and, therefore, developing advanced ROVs or smart ROVs for efficient operations in terms of operating hours and costs has received much attention. There are more than 550 AUVs in use worldwide and some of them are commercially available. While the use of AUVs is still limited, advantages of AUVs have been recognized by the recent AUV operations in the field: post-incident monitoring of BP Macondo oil leakage in 2010 and the successful recovery of black boxes of Air France flight 447 that crashed into the Atlantic Ocean in 2009. Therefore, it is expected to see more underwater robots for various applications in the future.

Some web-based resources related to the topics of this chapter include the following sites: the Remotely Operated Vehicles Committee of the Marine Technology Society (http://www.rov.org/), the Autonomous Undersea Vehicles Application Center (AUVAC, http://www.auvac.org) and the technical committee on Marine Robotics of the IEEE Society of Robotics and Automation (http://webuser.unicas.it/MarineRoboticsTC). A list of AUVs and developers can be found at http://www.transitport.net/Lists/AUVs.Org.html. AUVSI (Association for Unmanned Vehicle Systems International) and ONR (US Office of Naval Research) sponsor International Autonomous Underwater Vehicle Competition as well as Autonomous Surface Vehicle competition (http://www.auvsifoundation.org/AUVSI/FOUNDATION/Competitions/).

It is worthwhile to note that several important initiatives [25.1, 101] in underwater robotics in Europe have been made including AMADEUS for developing dexterous arms for underwater manipulation; GREX – coordination and control of cooperating heterogeneous unmanned systems in uncertain environments; CO-3AUVs – cognitive cooperative control for AUVs; FREESUBNET – a Marie Curie research training network on marine robotics; and TRIDENT that developed Girona 500 I-AUV for autonomous underwater intervention tasks.

Video-References

VIDEO 793 Six-legged waking underwater robot, Crabster
available from http://handbookofrobotics.org/view-chapter/25/videodetails/793

VIDEO 794 Preliminary results of sonar-based SLAM using landmarks
available from http://handbookofrobotics.org/view-chapter/25/videodetails/794

VIDEO 796 First record of deep-sea diving of Hamire, depth was 5882m
available from http://handbookofrobotics.org/view-chapter/25/videodetails/796

VIDEO 797 Preliminary experimental result of an ROV, iTurtle
available from http://handbookofrobotics.org/view-chapter/25/videodetails/797

VIDEO 799 Preliminary experimental result of an AUV, yShark2
available from http://handbookofrobotics.org/view-chapter/25/videodetails/799

References

25.1 J. Yuh, G. Marani, D.R. Blidberg: Applications of marine robotic vehicles, Intell. Serv. Robotics **4**(4), 221–231 (2011)

25.2 J. Yuh: Design and control of autonomous underwater robots: A survey, Auton. Robots **8**(1), 7–24 (2000)

25.3 S.W. Moore, H. Bohm, V. Jensen: *Underwater Robotics: Science, Design and Fabrication* (Marine Advanced Technology Education MATE Center, Monterey 2010)

25.4 R.D. Christ, R.L. Wernli: *The ROV Manual: A User Guide for Observation Class Remotely Operated Vehicles* (Elsevier, Amsterdam 2007)

25.5 K. Hardy, S. Weston, J. Sanderson: Under pressure: Testing before deployment is integral to success at sea, Sea Technol. **50**(2), 19–25 (2009)

25.6 T. Hyakudome: Design of autonomous underwater vehicle, Int. J. Adv. Robotic Syst. **8**(1), 131–139 (2011)

25.7 W.H. Wang, R.C. Engelaar, X.Q. Chen, J.G. Chase: The state-of-art of underwater vehicles – Theories and applications. In: *Mobile Robots – State of the Art in Land, Sea, Air, and Collaborative Missions*, ed. by X.Q. Chen, Y.Q. Chen, J.G. Chase (InTech, Rijeka 2009)

25.8 A.D. Bowen, D.R. Yoerger, C. Taylor, R. McCabe, J. Howland, D. Gomez-Ibanez, J.C. Kinsey, M. Heintz, G. McDonald, D.B. Peter, S.B. Fletcher, C. Young, J. Buescher, L.L. Whitcomb, S.C. Martin, S.E. Webster, M.V. Jakuba: The Nereus hybrid underwater robotic vehicle for global ocean science operations to 11,000 m depth, Proc. MTS/IEEE Ocean (2007)

25.9 T.J. Osse, T.J. Lee: Composite pressure hulls for autonomous underwater vehicles, Proc. MTS/IEEE Ocean (2007)

25.10 K. Hardy: Anodizing aluminum for underwater applications, Ocean News Technol. **15**(3), 54–56 (2009)

25.11 S.M.A. Sharkh, G. Griffiths, A.T. Webb: Power sources for unmanned underwater vehicles. In: *Technology and Applications of Autonomous Underwater Vehicles*, ed. by G. Griffiths (Taylor Francis, New York 2002) pp. 19–35

25.12 H. Yoshida: Fundamentals of underwater vehicle hardware and their applications. In: *Underwater Vehicles*, ed. by A.V. Inzartsev (InTech, Rijeka 2009) pp. 557–582

25.13 L.L. Whitcomb: Underwater robotics: Out of the research laboratory and into the field, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (2000) pp. 709–716

25.14 Ø. Hasvold, N.J. Størkersen, S. Forseth, T. Lian: Power sources for autonomous underwater vehicles, J. Power Sourc. **162**(2), 935–942 (2006)

25.15 A. Mendez, T.J. Leo, M.A. Herreros: Fuel cell power systems for autonomous underwater vehicles: State of the art, Proc. Int. Conf. Energy (2014)

25.16 K.L. Davies, R.M. Moore: Unmanned underwater vehicle fuel cell energy/power system technology assessment, IEEE J. Ocean Eng. **32**(2), 365–372 (2007)

25.17 H. Yoshida, T. Sawa, T. Hyakudome, S. Ishibashi, T. Tani, M. Iwata, T. Moriga: The high efficiency multi-less (HEML) fuel cell – A high energy source for underwater vehicles, buoys, and stations, Proc. MTS/IEEE Ocean (2011)

25.18 Q. Cai, D.J. Browning, D.J. Brett, N.P. Brandon: Hybrid fuel cell/battery power systems for underwater vehicles, Proc. 3rd SEAS DTC (2007)

25.19 K. E. Robinson: Li-poly pressure-tolerant batteries dive deep, Batter. Power Prod. Technol. **11**(2), 999999 (2007)

25.20 M.C. Wrinch, M.A. Tomim, J. Marti: An analysis of sub sea electric power transmission techniques from DC to AC 50/60 Hz and beyond, Proc. MTS/IEEE Ocean. (2007)

25.21 N. Størkersen, Ø. Hasvold: Power sources for AUVs, Proc. Sci. Def. Conf. (2004)

25.22 S. Cohan: Trends in ROV development, Mar. Technol. Soc. J. **42**(1), 38–43 (2008)

25.23 E. Mellinger: Power system for new MBARI ROV, Proc. IEEE Oceans (1993) pp. 152–157

25.24 M.R. Arshad: Recent advancement in sensor technology for underwater applications, Indian J. Mar. Sci. **38**(3), 267–273 (2009)

25.25 L. Lionel: Underwater robots part I: Current systems and problem pose. In: *Mobile Robotics*,

Avidbots Ex1016
Page 657 of 2154

ed. by A. Lazinica (InTech, Rijeka 2006) pp. 335–360
25.26 S.M.A. Sharkh: Propulsion systems for AUVs. In: *Technology and Applications of Autonomous Underwater Vehicles*, ed. by G. Griffiths (Taylor Francis, New York 2002) pp. 109–1255
25.27 T. Schilling, W. Klassen, C. Barrett, J. Stanley: Power at depth: Efficient ROV power delivery and thrust generation for improved construction, repair, and maintenance support, Proc. Offshore Technol. Conf. (2005)
25.28 P.D. Groves: *Principles of GNSS, Inertial, and Multisensor Integrated Navigation Systems* (Artech House, Boston 2013)
25.29 F. Viksten: *On the Use of an Accelerometer for Identification of a Flexible Manipulator*, Master Thesis (Linköping Univ., Linköping 2001)
25.30 J. Romeo, G. Lester: Navigation is key to AUV missions, Sea Technol. **42**(12), 24–30 (2001)
25.31 J.C. Kinsey, R.M. Eustice, L.L. Whitcomb: Aŝurvey of underwater vehicle navigation: Recent advances and new challenges, Proc. Int. Conf. Manoeuvering Control Mar. Craft, Lisbon (2006)
25.32 C. Silpa-Anan, T. Brinsmead, S. Abdallah, A. Zelinsky: Preliminary experiments in visual servo control for autonomous underwater vehicle, Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS) (2003) pp. 1824–1829
25.33 RTD Embedded Technologies, Inc., What is PC/104? http://www.rtd.com/PC104/Default.htm (2014)
25.34 A. Kenny, G. Lopez: Advances in and extended application areas for Doppler sonar, Proc. MTS/IEEE Ocean. (2012)
25.35 E. Thurman, J. Riordan, D. Toal: Multi-sonar integration and the advent of senor intelligence. In: *Advances in Sonar Technology*, ed. by S.R. Silva (InTech, Rijeka 2009) pp. 151–164
25.36 Sound Metrics Corporation (Bellevue, WA): Didson Sonar, L http://www.soundmetrics.com/products/didson-sonars (2015)
25.37 Y. Lee, T.G. Kim, H.-T. Choi: Añew approach of detection and recognition for artificial landmarks from noisy acoustic images, Adv. Intell. Syst. Comput. **274**, 851–858 (2014)
25.38 A. Alcocer, P. Oliveira, A. Pascoal: Underwater acoustic positioning systems based on buoys with GPS, Proc. 8th Europ. Conf. Underw. Acoust., Vol. 8 (2006) pp. 1–8
25.39 L. Brun: ROV/AUV trends market and technology, Mar. Technol. Rep. **5**(7), 48–51 (2012)
25.40 G. Verma, M. Kalra, S.K. Jain, D.A. Roy, B.B. Biswas: Embedded PC based controller for use in VME bus based data acquisition system, Proc. 9th Int. Workshop Pers. Comput. Part. Accel. Controls (2012) pp. 65–76
25.41 S.B. Williams, P. Newman, G. Dissanayake, J. Rosenblatt, H. Durrant-Whyte: Aďecoupled, distributed auv control architecture, Int. Symp. Robotics **31**, 246–251 (2000)
25.42 H.-T. Choi, A. Hanai, S.K. Choi, J. Yuh: Development of an underwater robot, ODIN-III. Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS) (2003) pp. 836–841
25.43 D. Lee, G. Kim, D. Kim, H. Myung, H. Choi: Vision-based object detection and tracking for autonomous navigation of underwater robots, Ocean Eng. **48**, 59–68 (2012)
25.44 Xilinx, Inc.: Zynq-7000 all programmable SoC, http://www.xilinx.com/products/silicon-devices/soc/zynq-7000/index.htm (2014)
25.45 B. Benson, Y. Li, R. Kastner, B. Faunce, K. Domond, D. Kimball, C. Schurgers: Design of a low-cost, underwater acoustic modem for short-range sensor networks, Proc. MTS/IEEE Ocean. (2010)
25.46 D.B. Kilfoyle, A.B. Baggeroer: The state of the art in underwater acoustic telemetry, IEEE J. Ocean. Eng. **25**(1), 4–27 (2000)
25.47 Z. Jiang: Underwater acoustic networks–issues and solutions, Int. J. Intell. Control Syst. **13**(3), 152–161 (2008)
25.48 M.W. Doniec, A. Xu, D. Rus: Robust real-time high definition underwater video streaming with AquaOptical II, Proc. IEEE Int. Conf. Robotics Autom. (ICRA), Karlsruhe (2013)
25.49 J.W. Nicholson, A.J. Healey: The present state of autonomous underwater vehicle (AUV) applications and technologies, Mar. Technol. Soc. J. **42**(1), 44–51 (2008)
25.50 W.D. Smart: Is ačommon middleware for robotics possible?, Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS) (2007)
25.51 K.P. Valavanis, D. Gracanin, M. Matijasevic, R. Kolluru, G.A. Demetriou: Control architectures for autonomous underwater vehicles, IEEE Control Syst. **17**(6), 48–64 (1997)
25.52 P. Ridao, J. Yuh, J. Batlle, K. Sugihara: On AUV control architecture, Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS) (2000) pp. 855–860
25.53 C. McGann, F. Py, K. Rajan, H. Thomas, R. Henthorn, R. McEwen: T-rex: Aḿodel-based architecture for AUV control, Proc. 3rd Workshop Plan. Plan Exec. Real-World Syst. (2007)
25.54 E.F. Perdomo, J.C. Gámez, A.C.D. Brito, D.H. Sosa: Mission specification in underwater robotics, J. Phys. Agents **4**(1), 25–34 (2010)
25.55 M.L. Seto (Ed.): *Marine Robot Autonomy* (Springer, New York 2013)
25.56 D. Goldberg: Huxley: Af̃lexible robot control architecture for autonomous underwater vehicles, Proc. MTS/IEEE Ocean. (2011)
25.57 B.S. Bingham, J.M. Walls, R.M. Eustice: Development of af̃lexible command and control software architecture for marine robotic applications, Mar. Technol. Soc. J. **45**(3), 25–36 (2011)
25.58 MOOS-IvP: http://oceanai.mit.edu/moos-ivp/pmwiki/pmwiki.php
25.59 M.R. Benjamin, H. Schmidt, P.M. Newman, J.J. Leonard: Nested autonomy for unmanned marine vehicles with MOOS-IvP, J. Field Robotics **27**(6), 834–875 (2010)
25.60 C. Lin, X. Feng, Y. Li, K. Liu: Toward ag̃eneralized architecture for unmanned underwater vehicles,

Avidbots Ex1016
Page 658 of 2154

Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (2011) pp. 2368–2373
25.61 N. Mohamed, J. Al-Jaroodi, I. Jawhar: Middleware for robotics: A survey, Proc. IEEE Conf. Robotics, Autom. Mechatron. (2008) pp. 736–742
25.62 M. Namoshe, N.S. Tlale, C.M. Kumile, G. Bright: Open middleware for robotics, Proc. 5th Int. Conf. Mechatron. Mach. Vis. Pract. (2008) pp. 189–194
25.63 D. Brugali, G.S. Broten, A. Cisternino, D. Colombo, J. Fritsch, B. Gerkey, G. Kraetzschmar, R. Vaughan, H. Utz: Trends in robotic software frameworks. In: *Software Engineering for Experimental Robotics*, ed. by D. Brugali (Springer, Berlin, Heidelberg 2007) pp. 259–266
25.64 A. Elkady, T. Sobh: Robotics middleware: A comprehensive literature survey and attribute-based bibliography, J. Robotics **2012**, 959013 (2012)
25.65 S. Kim, H.-T. Choi, J.-W. Lee, Y.J. Lee: Design, implementation, and experiment of an underwater robot for effective inspection of underwater structures, Proc. 2nd Int. Conf. Robot Intell. Technol. Appl. (2013)
25.66 T.W. Kim, J. Yuh, G. Marani: Underwater vehicle manipulators. In: *Springer Handbook of Ocean Engineering*, ed. by M. Dhanak, N. Xiros (Springer, Berlin, Heidelberg, 2016), in press.
25.67 N. Kato, D.M. Lane: Coordinated control of multiple manipulators in underwater robots, Proc. IEEE Int. Conf. Robotics Autom. (ICRA), Vol. 3 (1996) pp. 2505–2510
25.68 M.W. Dunnigan, D.M. Lane, A.C. Clegg, I. Edwards: Hybrid position/force control of a hydraulic underwater manipulator, Proc. IEEE Control Theory Appl. **143**, 145–151 (1996)
25.69 B. Lévesque, M.J. Richard: Dynamic analysis of a manipulator in a fluid environment, Int. J. Robotics Res. **13**(3), 221–231 (1994)
25.70 H. Mahesh, J. Yuh, R. Lakshmi: A coordinated control of an underwater vehicle and robotic manipulator, J. Robotics Syst. **8**(3), 339–370 (1991)
25.71 S. McMillan, D.E. Orin, R.B. McGhee: Efficient dynamic simulation of an underwater vehicle with a robotic manipulator, IEEE Trans. Syst. Man Cybern. **25**(8), 1194–1206 (1995)
25.72 T.W. McLain, S.M. Rock, M.J. Lee: Experiments in the coordinated control of an underwater arm/vehicle system, Autom. Robotics **3**, 213–232 (1996)
25.73 T.J. Tarn, G.A. Shoults, S.P. Yang: A dynamic model of an underwater vehicle with a robotic manipulator using kanes method, Autom. Robot. **3**, 269–283 (1996)
25.74 K. Ioi, K. Itoh: Modelling and simulation of an underwater manipulator, Adv. Robot. **4**(4), 303–317 (1989)
25.75 I. Schjølberg, T.I. Fossen: Modelling and control of underwater vehicle-manipulator systems, Proc. 3rd Conf. Mar. Craft Manoeuvering Control (1994)
25.76 K.N. Leabourne, S.M. Rock: Model development of an underwater manipulator for coordinated arm-vehicle control, Proc. MTS/IEEE Ocean., Vol. 2 (1998) pp. 941–946
25.77 M. Lee, H.-S. Choi: A robust neural controller for underwater robot manipulators, IEEE Trans. Neural Netw. **11**(6), 1465–1470 (2000)
25.78 J.-H. Ryu, D.-S. Kwon, P.-M. Lee: Control of underwater manipulators mounted on an ROV using base force information, Proc. IEEE Int. Conf. Robotics Autom. (ICRA), Vol. 4 (2001) pp. 3238–3243
25.79 M.H. Patel: *Dynamics of Offshore Structures* (Butterworths, London 1989)
25.80 A.W. Troesch, S.K. Kim: Hydrodynamic forces acting on cylinders oscillating at small amplitudes, J. Fluids Struct. **5**(1), 113–126 (1991)
25.81 M. Hildebrandt, L. Christensen, J. Kerdels, J. Albiez, F. Kirchner: Realtime motion compensation for ROV-based teleoperated underwater manipulators, Proc. MTS/IEEE Ocean. Eur. (2009) pp. 1–6
25.82 O. Brock, R. Grupen: *Final Report of NSF/NASA Workshop on Autonomous Mobile Manipulation (AMM)* (Univ. of Massachusetts, Amherst 2005)
25.83 G. Antonelli: *Underwater Robots*, Springer Tracts in Advanced Robotics, Vol. 96, 3rd edn. (Springer, Berlin, Heidelberg 2014)
25.84 M. Carreras, J. Yuh, J. Batlle, P. Ridao: A behavior-based scheme using reinforcement learning for autonomous underwater vehicles, IEEE J. Ocean. Eng. **30**(2), 416–427 (2005)
25.85 M. Carreras, J. Yuh, J. Batlle, P. Ridao: Application of SONQL for real-time learning of robot behaviors, Robots Auton. Syst. **55**(8), 628–642 (2007)
25.86 S. Zhao, J. Yuh: Experimental study on advanced underwater robot control, IEEE Trans. Robotics **21**(4), 695–703 (2005)
25.87 A. Hanai, H.-T. Choi, S.K. Choi, J. Yuh: Experimental study on fine motion control of underwater robots, Adv. Robotics **18**(10), 963–978 (2004)
25.88 T.W. Kim, J. Yuh: Application of on-line neuro-fuzzy controller to AUVs, Inf. Sci. **145**(1), 169–182 (2002)
25.89 C.S.G. Lee, J.-S. Wang, J. Yuh: Self-adaptive neuro-fuzzy systems for autonomous underwater vehicle control, Adv. Robotics **15**(5), 589–608 (2001)
25.90 J. Yuh, J. Nie: Application of non-regressor-based adaptive control to underwater robots: Experiment, Comput. Electr. Eng. **26**(2), 169–179 (2000)
25.91 K.C. Yang, J. Yuh, S.K. Choi: Fault-tolerant system design of an autonomous underwater vehicle ODIN: An experimental study, Int. J. Syst. Sci. **30**(9), 1011–1019 (1999)
25.92 H.H. Wang, S.M. Rock, M.J. Lees: Experiments in automatic retrieval of underwater objects with an AUV, Proc. MTS/IEEE Ocean., Vol. 1 (1995) pp. 366–373
25.93 J. Evans, P. Redmond, C. Plakas, K. Hamilton, D. Lane: Autonomous docking for intervention-AUVs using sonar and video-based real-time 3D pose estimation, Proc. MTS/IEEE Ocean., Vol. 4 (2003) pp. 2201–2210

Avidbots Ex1016
Page 659 of 2154

25.94 G. Marani, S.K. Choi, J. Yuh: Underwater autonomous manipulation for intervention missions AUVs, Ocean Eng. **36**(1), 15–23 (2009)
25.95 G. Marani, J. Yuh, S.K. Choi: Autonomous manipulation for an intervention AUV. In: *Advances in Unmanned Marine Vehicles*, IEE Control Engineering Series, ed. by B. Sutton, G. Roberts (Institution of Engineering and Technology, London 2006) pp. 217–237
25.96 G. Marani, S.K. Choi, J. Yuh: Real-time center of buoyancy identification for optimal hovering in autonomous underwater intervention, Intell. Serv. Robotics **3**(3), 175–182 (2010)
25.97 D. Beciri: SAUVIM robot completed its first fully autonomous mission, http://www.robaid.com/search/SAUVIM (2010)
25.98 P. Sanz, R. Ridao, G. Oliver, P. Casalino, C. Insaurralde, C. Silvestre, C. Melchiorri, A. Turetta: TRIDENT: Recent improvements about autonomous underwater intervention missions, Proc. IFAC Workshop Navig. Guid. Control Underw. Veh. (NGCUV) (2012)
25.99 D. Ribas, N. Palomeras, P. Ridao, M. Carreras, A. Mallios: Girona 500 AUV: From survey to intervention, IEEE/ASME Trans. Mechatron. **17**(1), 46–53 (2012)
25.100 M. Prats, D. Ribas, N. Palomeras, J.C. Garcia, V. Nannen, S. Wirth, J.J. Fernañdez, J.P. Beltrañ, R. Campos, P. Ridao, P.J. Sanz, G. Oliver, M. Carreras, N. Gracias, R. Marín, A. Ortiz: Reconfigurable AUV for intervention missions: Ačase study on underwater object recovery, Intell. Serv. Robotics **5**(1), 19–31 (2012)
25.101 Trident: Marine Robots and Dexterous Manipulation for Enabling Autonomous Underwater Multipurpose Intervention Missions: Newsletter October 2012, http://www.irs.uji.es/trident/files/2nd-TRIDENT-SCHOOL-Newsletter-Oct2012.pdf (2012)

Avidbots Ex1016
Page 660 of 2154

Multimedia Contents

26. Flying Robots

Stefan Leutenegger, Christoph Hürzeler, Amanda K. Stowers, Kostas Alexis, Markus W. Achtelik, David Lentink, Paul Y. Oh, Roland Siegwart

Unmanned aircraft systems (UASs) have drawn increasing attention recently, owing to advancements in related research, technology, and applications. While having been deployed successfully in military scenarios for decades, civil use cases have lately been tackled by the robotics research community.

This chapter overviews the core elements of this highly interdisciplinary field; the reader is guided through the design process of aerial robots for various applications starting with a qualitative characterization of different types of UAS. Design and modeling are closely related, forming a typically iterative process of drafting and analyzing the related properties. Therefore, we overview aerodynamics and dynamics, as well as their application to fixed-wing, rotary-wing, and flapping-wing UAS, including related analytical tools and practical guidelines. Respecting use-case-specific requirements and core autonomous robot demands, we finally provide guidelines to related system integration challenges.

26.1 **Background and History** 624
26.1.1 A Glimpse of History 624

26.2 **Characteristics of Aerial Robotics** 625
26.2.1 Aerial Robots Classification 625
26.2.2 The Effect of Scale 626

26.3 **Basics of Aerodynamics and Flight Mechanics** 629
26.3.1 Properties of the Atmosphere 630
26.3.2 General Fluid Dynamics and 2-D Flow around Airfoils 630
26.3.3 Wing Aerodynamics 633
26.3.4 Performance of Rotors and Propellers 635
26.3.5 Drag 637
26.3.6 Aircraft Dynamics and Flight Performance Analysis 638

26.4 **Airplane Modeling and Design** 641
26.4.1 Forces and Moments 641
26.4.2 Static Stability 642
26.4.3 Dynamic Model 643
26.4.4 Design Guidelines 644
26.4.5 A Simple Autopilot 646

26.5 **Rotorcraft Modeling and Design** 647
26.5.1 Mechanical Design of Rotors and Propellers 648
26.5.2 Rotorcraft Dynamics 648
26.5.3 Simplified Aerodynamics 649
26.5.4 Nonuniform Inflow 651
26.5.5 Flapping Dynamics 651
26.5.6 Flight Dynamics Assessment 652

26.6 **Flapping Wing Modeling and Design** 653
26.6.1 Aerodynamic Mechanisms 653
26.6.2 Sizing New Flappers 655

26.7 **System Integration and Realization** 659
26.7.1 Challenges for Autonomous UAS 659
26.7.2 Levels of Autonomy 659
26.7.3 UAS Components 660

26.8 **Applications of Aerial Robots** 662
26.8.1 Demonstrated Applications of UAS 662
26.8.2 Current Applications and Missions 664
26.8.3 Aerial Robots: Emerging Categories 664
26.8.4 Open Issues 665

26.9 **Conclusions and Further Reading** 666

Video-References 666

References 667

Avidbots Ex1016
Page 661 of 2154

26.1 Background and History

The field of *aerial robotics* encompasses a very broad class of flying machines that nowadays often possess the perception capabilities and decisional autonomy to accomplish complex tasks without the need for any direct human interventioning. Historically and within the aerospace jargon, robotic flying machines are commonly referred to as *unmanned aerial vehicles* (UAVs), while the entire infrastructures, systems and human–machine interfaces required for autonomous operation are often called *unmanned aerial systems* (UAS). Aerial robotic technologies are currently on the cutting edge of aerospace and robotic research. Breakthrough contributions take place in various fields such as design, estimation [26.1], perception [26.2], control [26.3], and planning [26.4], paving the way for a historical change on how flying systems are operated and what application challenges they fulfill.

As a class of systems, aerial robots have their roots in the first guided missiles; however, nowadays they refer to a wide variety of advanced intelligent systems. According to the American Institute of Aeronautics and Astronautics (AIAA) [26.5], a UAV is defined as

> *an aircraft which is designed or modified, not to carry a human pilot and is operated through electronic input initiated by the flight controller or by an onboard autonomous flight management control system that does not require flight controller intervention.*

As is generally the case in robotics, aerial robots tend to become more and more complex systems as a result of the effort to achieve advanced decision making and planning capabilities based on its on-board perception of the environment and a set of relatively abstract mission goals.

Aerial robots posses the unique capability to gently fly over terrain that other robots struggle to roll or crawl over. The price to be paid is related with the advanced challenges in terms of system design, propulsion, perception, control, and navigation. Autonomous flight requires handling of all six degrees of freedom and advanced cognition capabilities within challenging environments. In that sense, perception and navigation complexity drastically increase, while payload and available power consumption for processing tends to be limited, especially as scale decreases. Essentially, the design of aerial robots requires increased attention and thorough selection, or even combination, of one or more existing or new flying concepts, electronic components and algorithms. The design engineer has to assess specific optimization challenges and trade-offs as important desired goals like decreased weight and modularity typically contradict each other.

26.1.1 A Glimpse of History

Aerial robotics is a field of active research and promising perspectives, yet it already accumulates more than a century of developments. Figure 26.1 depicts some historical as well as recent examples of UAVs in the military and civilian sector. Starting as conceptual designs in the context of the human efforts to develop flying machines, aerial robots soon proved their extensive potential and have already created their own legacy. As was also the case for manned aviation, aerial robotic technologies accelerated within the framework of the 20th century world conflicts. Within World War I, Hewitt–Sperry developed an automatic plane that acted as a flying torpedo, carrying onboard intelligence to autonomously sustain flight over long periods of time. This page-turning success was achieved through the integration of (Sperry's self-made) gyroscopes which were then mechanically connected to the control surfaces and therefore established the necessary feedback control loop. During World War II, the German armed forces deployed one of the first successful cruise missiles, the V–1. Despite the fact that V–1 had limited success rate it did incorporate most of the elementary components, estimation algorithms and control loops that can allow autonomous navigation and reference tracking. Military applications kept being, and still are, the main driving force of aerial robotics research and the newest developments in the area change and shape the modern warfare. With the introduction of global positioning systems (GPSs), aerial robots managed to achieve the first completely autonomous surveillance missions. As information and intelligence gathering became one of the most important aspects of the world's open or silent conflicts, military research around the 1970s led to systems equipped with cameras and other sensory systems, giving birth to the UAV prototype the way we know it today. However, civil applications are currently emerging at a very fast pace and the majority of market predictions converge to the conclusion that this area will take dominant characteristics, and most importantly, will become an equally important – if not more – innovation drive.

Within this framework, the advancements in the field of microprocessors, miniaturized sensing, as well as actuator efficiency and downscaling greatly accelerated the field of aerial robots and paved the way for the great achievements we observe today. Aerial robots have advanced to a state in which sophisticated sensor

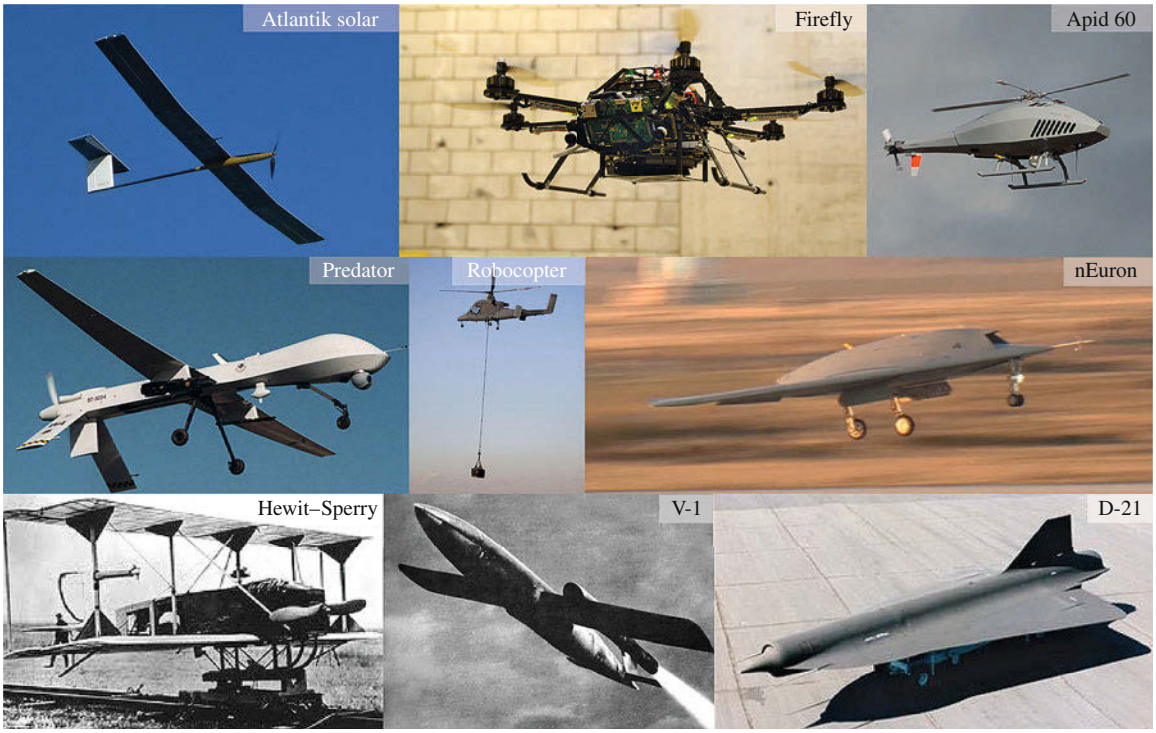


Fig. 26.1 A glimpse on the UAS history through some examples starting from the Hewitt–Sperry Automatic Airplane (1917), the V-1 flying bomb (1944), and the Lockheed D-21 (1962) until the recent examples of military (Predator, Robocopter, nEuron) and civilian (AtlantikSolar, Firefly, Apid 60) aerial robots

modules for onboard state estimation and environmental perception, powerful embedded processors running sophisticated navigation algorithms, potentially several communication interfaces, as well as high-end-mission-oriented payloads that enable the execution of challenging tasks, can be tightly integrated.

26.2 Characteristics of Aerial Robotics

This section aims to provide an overview of the key characteristic features and properties of different aerial robotic configurations as well as a classification based on the key advantages and limitations of some of the most common flying concepts found in unmanned aviation.

26.2.1 Aerial Robots Classification

Compared to the categorization of manned aviation, aerial robots classification is more complex, as the term currently refers to a very wide variety of systems of different scale, mechanical configuration, and actuation principles. In their vast majority, aerial robots correspond, in one way or another, to miniaturized versions of manned aircraft designs. Relatively classical fixed-wing unmanned aerial systems (FW-UAS) designs and rotary-wing unmanned aerial systems (RW-UAS) such as those shown in Fig. 26.2 are common vehicle configurations one may encounter in most applications, including those of surveillance, monitoring, inspection, mapping, or payload transportation. However, even within these relatively traditional concepts, several design aspects differ from those chosen for manned systems. This reflects the fact that for different scales, the variation of the physical properties behavior, along with the search for optimized designs, will naturally lead to modified and novel design considerations. This is further triggered by the fact that the absence of a pilot on-board unlocks a wide set of engineering choices, typically out of question or even forbidden in manned aviation. As expected for a multitude of engineering reasons, large UAS tend to follow design concepts closer to – while at smaller scale to classi-

Avidbots Ex1016
Page 663 of 2154

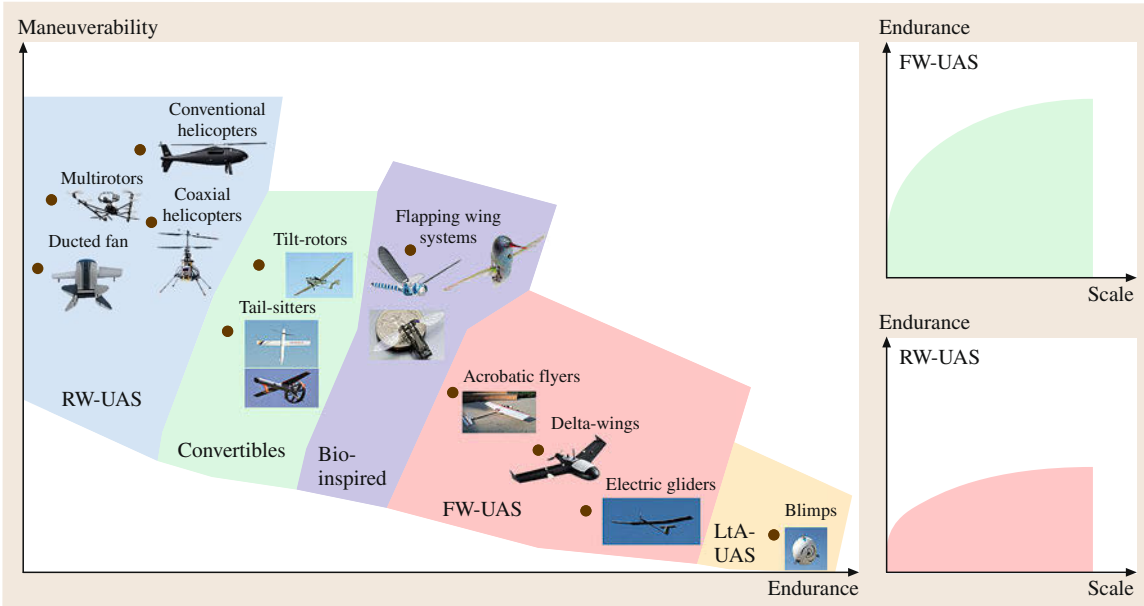


Fig. 26.2 Classification of aerial robotics based on their endurance and maneuverability properties. Also note the significant effect of scale which highlights that comparisons should be done on similar scales

cal designs while as scale decreases innovation – at the level of the flying principle – becomes more and more intense.

Apart from *lighter-than-air* systems (LtA-UAS), FW-UAS tend to be the most power efficient flying principle, while RW-UAS are tailored to increased maneuverability as well as the ability of stationary vertical flight (*hovering*). This general classification (also valid for manned aviation) is then further complicated with the relatively large class of convertible designs (such as tilt-rotors or cruise-flight-enabled ducted fans). This first attempt for aerial robots classification has then to be further augmented to account for the biologically inspired concepts, and especially the emerging field of flapping-wing UAS (FI-UAS). Figure 26.2 provides an abstract – yet incomplete – overview of the vehicle classes one may encounter in most of the application fields. As shown, a large diversity is observed as a result of the engineering efforts to propose designs with optimized endurance, agility, controllability, or even simplicity in a very wide scale range. In the following subsections, a brief overview on how the main aerodynamic forces and effects depend on the design scale of an aerial vehicle are provided.

26.2.2 The Effect of Scale

The understanding of how aerial vehicles manage to remain airborne, provides a useful insight into the effect of scale, and how different dimensioning has a huge impact on the efficiency of every flying machine. Table 26.1 provides an overview of the formulas expressing the lift force, as well as the drag forces that govern the flight of the most common UAS configurations. More detailed definitions on the aerodynamic forces can be found on the subsequent sections.

Within these equations, ρ is the density of the air while the remaining parameters are specific to the vehicle configuration. For FW-UAS, c_L and c_D represent the wing lift and drag coefficients, respectively, A is the wing area, and V_t denotes the airspeed. For the case of RW-UAS, c_T and c_Q denote the rotor thrust and drag coefficients, (πR^2) is the rotor disk area, Ω is the angular velocity of the rotor, and R is the rotor disk radius. Finally, for LtA-UAS, V^{LtA} is the volume of the blimp, c_D^{LtA} is the drag coefficient depending on the blimp shape, V_t is the blimp's airspeed, A^{LtA} is the blimp surface in the direction of motion and ρ_{gas} is the filling gas density. Figure 26.3 illustrates

Table 26.1 Formulas of the main aerodynamic forces and moments for common UAS configurations. FW stands for *fixed-wing* UAS, RW for *rotary-wing*, and LtA for *lighter-than-air*

UAS	Lift/Thrust	Drag force/Moment
FW	$L = \frac{1}{2} c_L \rho A V_t^2$	$D = \frac{1}{2} c_D \rho A V_t^2$
RW	$T = c_T \rho (\pi R^2)(R\Omega)^2$	$Q = c_Q \rho (\pi R^2)(R\Omega)^2 R$
LtA	$L_s = -g V^{LtA}(\rho_{gas} - \rho)$	$D = \frac{1}{2} c_D^{LtA} \rho A^{LtA} V_t^2$

Avidbots Ex1016
Page 664 of 2154

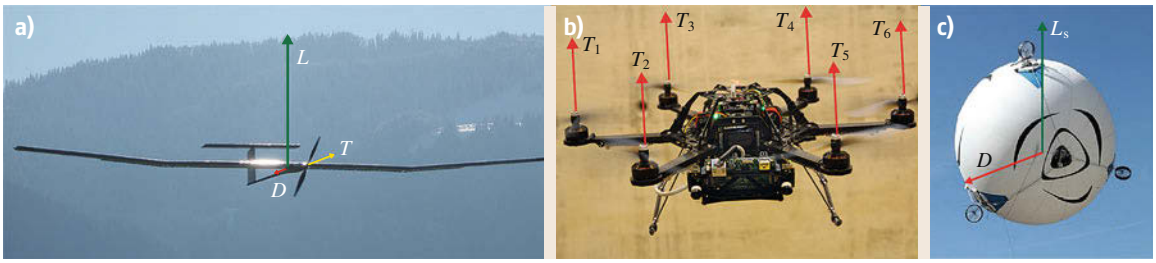


Fig.26.3a–c Main aerodynamic forces applied on **(a)** fixed-wing, **(b)** rotary-wing, and **(c)** lighter-than-air systems. **(a)** AtlantikSolar is a solar-powered FW-UAS developed by the Autonomous Systems Lab at ETH Zurich, **(b)** Firefly is developed by Ascending Technologies GMBH, while **(c)** Skye is developed by students of ETH Zurich

these forces on the body of the relevant aerial vehicle configurations.

Derivation of *scaling laws* starts with the observation of the lift and drag forces and how these are functions of scale-dependent parameters such as the area of the wing or the rotor radius. Proper dimensioning is essentially a very complex procedure where a multitude of factors has to be taken into account. Among others, one has to account for the issues of aerodynamics efficiency, availability of propulsion systems at a given scale, the technologies they employ (e.g., electric motors, jet engines) as well as the simplicity and robustness of the corresponding mechanical configuration. In the following, scaling laws and relevant design guidelines for fixed-wing, rotary-wing, and lighter-than-air systems are provided. Only a brief overview is provided for the case of flapping-wing systems, as the effect of scale on such UAS configurations is separately discussed within Sect. 26.6.

FW-UAS

Scaling laws express the dominant role of size and scale for a given vehicle configuration. In the case of fixed-wing systems, the wing loading, defined as the ratio of the weight (W) versus the wing area A, is the key parameter one has to focus to get some first insight on the role of scale. The Tennekes diagram shown in Fig. 26.4 provides a visual interpretation of this fact [26.6]. Working around the point that the lift force exactly counteracts the weight, the indicated trend line was derived using the following formulas [26.7]

$$\frac{W}{A} = \sqrt[3]{W}47\,,$$
$$\frac{W}{A} = \frac{1}{2}c_{\mathrm{L}}\rho V_{\mathrm{t}}^2\,,$$
$$A = b_{\mathrm{w}}c_{\mathrm{w}}\,, \qquad (26.1)$$

where V_{t} is the airspeed, W is the weight, A is the wing area, b_{w} is the wing span, and c_{w} is the wing chord. These equations express the role of the lifting properties of the airfoil and airspeed against the ratio of the weight of the flying body and its wing area. For this analysis, a fixed aspect ratio ($\Lambda = b_{\mathrm{w}}/c_{\mathrm{w}}$) is assumed for all sizes of aircraft. Although such a simple analysis does not account for the details of the fluid dynamics environment between the different aircraft sizes, it is known that smaller aircrafts are typically built with lower aspect ratios, and that the difference in aspect ratio over existing aircraft within the size range of interest is significant.

RW-UAS

For the case of rotorcraft configurations, similar scaling laws regarding the vehicle efficiency may be derived. It is important to highlight however that especially for rotorcrafts, working with scaling laws demands that one has to simultaneously focus on both efficiency and dynamic response in order to avoid undesired effects in the vehicle flight dynamics such as unstable oscillations. Regarding the power efficiency, let power loading (PL) be defined as T/P, where P corresponds to the ideal power. As the induced ideal power to hover is given by $P = T\upsilon_{\mathrm{h}}$, the *ideal power loading* will be inversely proportional to the induced velocity at the rotor disk υ_{i}

$$\upsilon_{\mathrm{h}} = \upsilon_{\mathrm{i}} \Rightarrow \sqrt{\frac{T}{2\rho(\pi R^2)}} = \frac{P}{T} = (PL)^{-1}\,. \qquad (26.2)$$

Observing Fig. 26.5, it is shown that the ratio T/P decreases quickly with increasing disk loading. Therefore, configurations with proportionally smaller rotors against their mass will tend to be less efficient in hovering flight; that is, the rotor will require proportionally more power to generate the required amount of thrust. It is also to be noted, however, that calculation of the actual power loading and rotor efficiency requires the consideration of viscous losses.

From the above brief analysis, we concluded that in general the tendency to increase the rotor dimension favors efficiency. However, this is not the only scaling law one has to consider. Rotorcrafts are particularly com-

Avidbots Ex1016
Page 665 of 2154

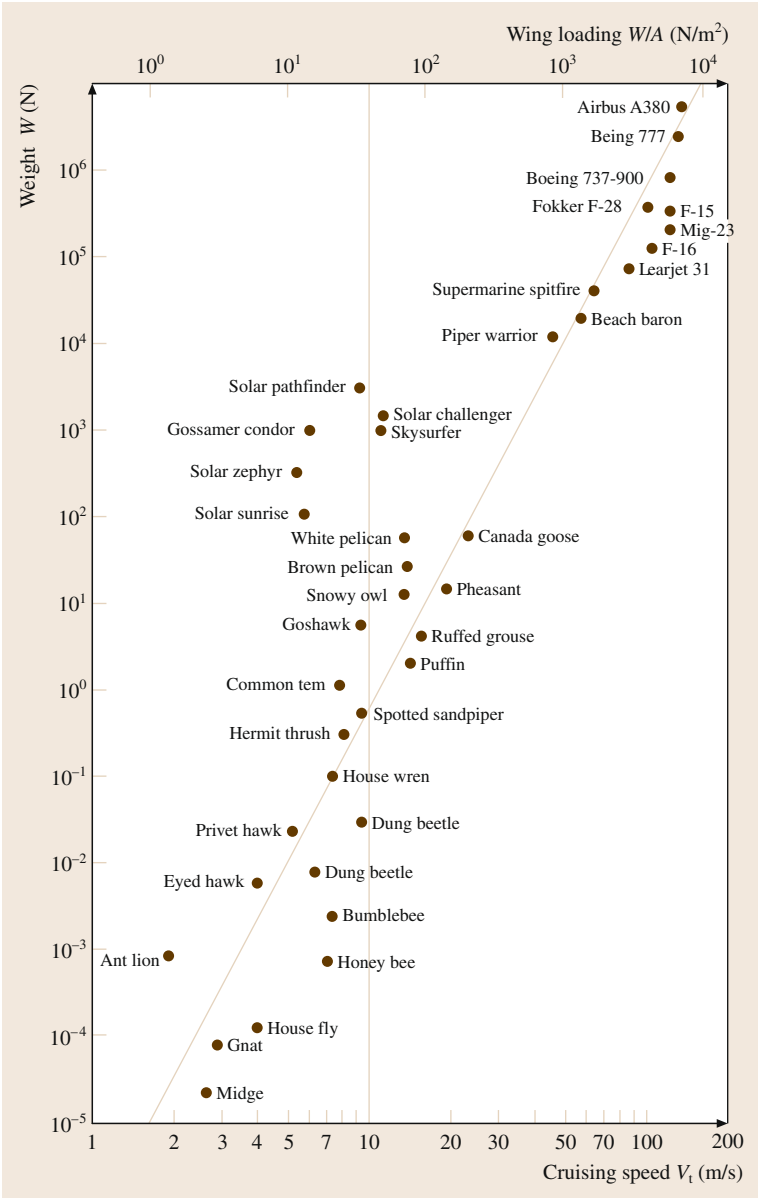


Fig. 26.4 Tennekes size trend relating wing loading and cruise speed to weight for insects, birds, and manned aircraft

plex dynamic systems and scaling considerations also have to focus on dynamic aspects of their flight. A more concrete analysis may take place using Froude or Mach scaling models. Let N denote the length scale between two vehicles, R_m the rotor radius of the model vehicle, and R_p the rotor radius of the prototype vehicle: thus, a scale factor N denotes a helicopter $1/N$ times the size of its prototype. Table 26.2 summarizes the Froude and Mach scaling laws that account for the role of scale in a set of significant parameters namely the length of the model and the prototype L_m, L_p, the dominant time constants t_m, t_p of the inner-loop characteristic response, the characteristic velocities V_m, V_p, the weight values W_m, W_p, the expected moments of inertia I_m, I_p, and the response-dominant frequencies ω_m, ω_p.

These, slightly more advanced scaling laws, further provide the opportunity to assess the aspects of main rotor performance, and more specifically the expected thrust margin. Traditional manned helicopters have small thrust margins in hover, typically 5–10% while miniaturized vehicles often present very high values. Mach models predict in general faster rotor speeds as compared to Froude scaled models, which consequently leads to a lower expected thrust coefficient. The

Avidbots Ex1016
Page 666 of 2154

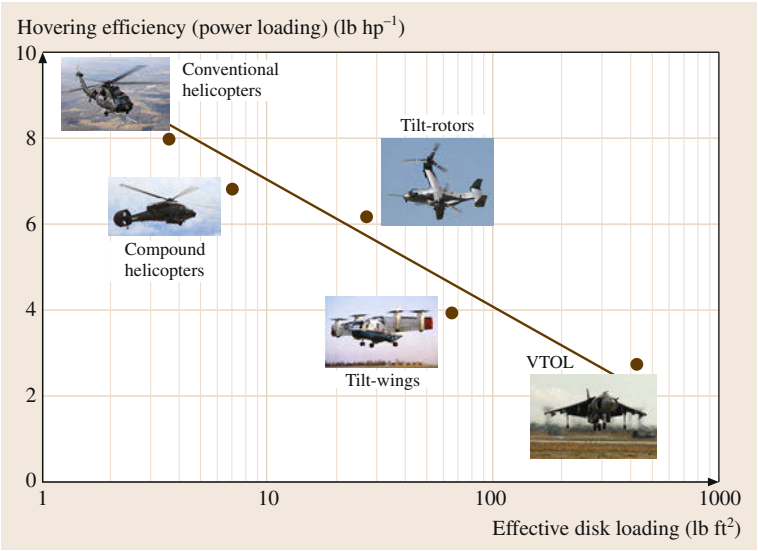


Fig. 26.5 Hovering efficiency versus disk loading for a range of vertical lift aircraft

Table 26.2 Scaling laws for conventional helicopters

Dimension	Froude	Mach
Length	$L_m = L_p/N$	$L_m = L_p/N$
Time constant	$t_m = t_p/\sqrt{N}$	$t_m = t_p/N$
Speed	$V_m = V_p/\sqrt{N}$	$V_m = V_p$
Weight	$W_m = W_p/N^3$	$W_m = W_p/N^3$
Inertia mat.	$I_m = I_p/N^5$	$I_m = I_p/N^5$
Frequency	$\omega_m = \omega_p\sqrt{N}$	$\omega_m = \omega_p N$

thrust coefficient reflects the lift loading of the rotor. For a given, single rotor configuration, the maximum thrust is provided by the following expression

$$T_{max} = (c_T/\sigma)_{max}\rho(\pi R^2)(\Omega R)^2\,, \qquad (26.3)$$

where σ represents the blade solidity (Sect. 26.3.4). This relation gives a maximum thrust that scales as $T_{max} \propto 1/N^3$ for a Froude model and as $T_{max} \propto 1/N^2$ for a Mach model. Once divided by the vehicle weight which scales as $W \propto 1/N^3$, it is deduced that Froude models present a similar thrust-to-weight ratio. On the contrary, for a Mach model, there is an increasing expected maximum thrust-to-weight: $(T/W)_{max} \propto N$. Using these formulas, researchers in [26.8] calculated the scaling parameters for several conventional helicopters that provide intuitive insight on how scaling laws work.

LtA-UAS

For the case of lighter-than-air vehicles simple scaling laws regarding the efficiency of the system hold. Considering the example of a spherical blimp, it is directly deduced that the lift force scales with the cubic power of the radius. On the other hand, its mass, which depends on the surface, scales with the square power of the radius and also does the drag force. This essentially indicates that larger blimps will tend to have a higher maximum lift to weight and lift-against-drag force ratios.

FI-UAS

Analysis of the scaling laws for flapping wing systems requires a different treatment, as the flight modality changes while the robot operates in *hover* mode or navigates in *forward flight*. Furthermore, the lift and drag coefficients are dependent on the airfoil characteristics of the wing and also on the flapping frequency, a fact that further increases the complexity of the analysis on the effect of scale. Section 26.6 provides insight on how to deal with this challenging issue so that proper flapping wing systems design is achieved.

26.3 Basics of Aerodynamics and Flight Mechanics

Assembling an analytic representation of a UAS involves the derivation of approximative expressions for the aerodynamic forces, accounting for the actuator dynamics and appending the resulting effect to the vehicle body equations of motion. The goal of this section is to provide the necessary insight and understanding of

Avidbots Ex1016
Page 667 of 2154

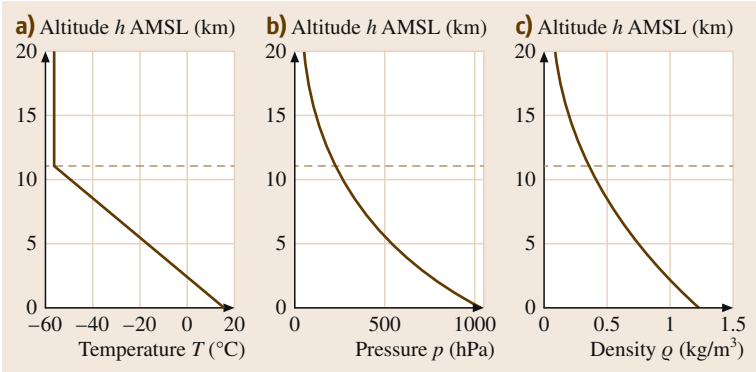


Fig. 26.6 Variation of **(a)** air temperature, **(b)** pressure, and **(c)** density with altitude in the lower part of the International Standard Atmosphere.The tropopause, above which the temperature is not further decreasing, corresponds to the red dashed line. It constitutes the upper limit of common weather phenomena

the underlying mechanisms and physical phenomena along with the derivation of the formulae for the most dominant effects one has to account for with any UAS configuration.

26.3.1 Properties of the Atmosphere

Assessing flow properties forms the basis for any further qualitative or quantitative aerodynamic analysis relevant for aircraft design, modeling and control. The international standard atmosphere (ISA) [26.9] provides a reference for the average main air characteristics as a function of altitude. Figure 26.6 shows the evolution of air temperature T_{air}, pressure p, and density ρ. These parameters largely affect the Reynolds number Re, which can be interpreted as the influence of inertial forces as compared to viscous forces of a flow, as well as the Mach number Ma representing the ratio of airspeed versus speed of sound.

It is noteworthy that the above parameters may be brought into relationship by the ideal gas law

$$p = \rho R T_{\text{air}} \,, \tag{26.4}$$

with $R = 286.97\,\text{m}^2/\text{s}^2/^\circ\text{K}$ denoting the ideal gas constant of air.

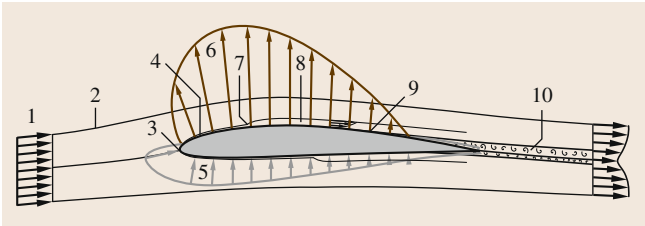


Fig. 26.7 Characteristics of a 2-D flow around an airfoil 1 – Free stream velocity field, 2 – Streamline, 3 – Stagnation point, 4 – yLaminar boundary layer, 5 – Overpressure, 6 – ySuction, 7 – Transition point, 8 – Turbulent boundary layer, 9 – Separation point, 10 – Separated flow

In contrast to *aerodynamics*, where forces, first and foremost lift, is generated by motion of an object through the air, the *aerostatic* lift force is formed solely by static properties of an object. It forms the basis of operation for a balloon or blimp.

According to Archimedes' principle, the aerostatic lift L_{stat} pointing upward, amounts to

$$L_{\text{stat}} = \rho V g - m g \,, \tag{26.5}$$

where g stands for the Earth gravitational acceleration, V for the volume of the object, and m for its mass. To state an example, consider a helium balloon of spherical shape with a diameter amounting to 1 m in the lower atmosphere. Neglecting its hull weight (thus with $m = \rho_{\text{helium}} V$), it will generate an aerostatic lift force of 5.4 N, representing an upper bound for any kind of total design mass including payload. Increasing the diameter of said sphere to 1.5 m has the huge effect of increasing lift to 18.1 N.

26.3.2 General Fluid Dynamics and 2-D Flow around Airfoils

A general airflow around an aircraft is three-dimensional, unsteady, and may be turbulent, even interacting with a nonrigid structure. In this setting, computations are almost intractable. We will thus rely on certain simplifications in order to allow simpler calculations and notably enhancing the understanding of a flow field together with resulting forces and moments. In both airplane and rotorcraft aerodynamics, the assumption of a locally two-dimensional (2-D) flow can be very helpful to serve as a starting point for more advanced computation.

Before diving into a formal treatment, the example in Fig. 26.7 depicts some characteristic elements of 2-D flow around an airfoil.

Notice that the pressure distribution on the airfoil contour is induced by the flow field, making the most

significant contribution to the aerodynamic force and moment. However, also viscous effects yield a typically unwanted share in the overall force (and moment) in the form of shear stress transmitted to the surface. The elements overviewed in Fig. 26.7 will now be explained in the following paragraphs.

Finite Control Volume Analysis: Mass and Momentum Conservation

Consider a finite control volume B bounded by the surface S with normal $\boldsymbol{n}$, which may contain a body or airfoil, depending on the context. For convenience, parts of the boundaries are often chosen to be streamlines (in 2-D) or stream surfaces (in 3-D). For this volume, the *conservation of mass* must be fulfilled

$$\iint_S \rho \boldsymbol{v} \cdot \boldsymbol{n} \, \mathrm{d}S \,, \tag{26.6}$$

where $\boldsymbol{n}$ denotes the fluid velocity vector.

From classical Newtonian mechanics, we can furthermore postulate the applicability of *conservation of linear and angular momentum*

$$\boldsymbol{F}_{\mathrm{tot}} = \iint_S \rho \boldsymbol{v} (\boldsymbol{v} \cdot \boldsymbol{n}) \, \mathrm{d}S + \iint_S p \boldsymbol{n} \, \mathrm{d}S + \frac{\partial}{\partial t} \iiint_B \rho \boldsymbol{v} \, \mathrm{d}B \,, \tag{26.7}$$

$$\boldsymbol{M}_{\mathrm{tot}} = \iint_S \rho (\boldsymbol{v} \times \boldsymbol{r}) (\boldsymbol{v} \cdot \boldsymbol{n}) \, \mathrm{d}S + \frac{\partial}{\partial t} \iiint_B \rho (\boldsymbol{v} \times \boldsymbol{r}) \, \mathrm{d}B \,, \tag{26.8}$$

where $\boldsymbol{r}$ denotes the position vector.

Differential Volume Analysis: Euler and Bernoulli Equations

When applied to a differential volume and assuming inviscid flow, (26.7) can be used to derive *Euler's equation*

$$\rho \left(\frac{\partial}{\partial t} + \boldsymbol{v} \cdot \nabla \right) \boldsymbol{v} + \nabla p = \boldsymbol{0} \,. \tag{26.9}$$

This equation forms the basis of many finite-element-based numerical tools neglecting viscous effects outside the boundary layer. Such methods employ potential flow theory along with some boundary layer analysis module; free example tools are JavaFoil [26.10] and xfoil [26.11] for 2-D flow computation as well as XFLR [26.12] allowing also 3-D flow extensions.

When applying (26.9) along a streamline and under the assumption that the flow is incompressible (a fair assumption for low-speed aerodynamics up to Ma = 0.3), *Bernoulli's equation* relating speed (V_{t}) and pressure can be formulated

$$\rho \frac{V_{\mathrm{t}}^2}{2} + \rho g h + p = \mathrm{const} \,, \tag{26.10}$$

where g denotes gravitational acceleration and h the elevation – the aerostatic pressure component $\rho g h$ can often be neglected due to small elevation changes along a streamline.

Viscous Effects and the Boundary Layer

While it is often a valid approximation to neglect viscous effects far enough from the body surfaces, they have to be considered within the boundary layer, where the fluid is slowed down to meet the speed of the surface. The friction shear stress τ_{w} transmitted to the surface is characterized by the gradient of the flow speed perpendicular to the surface

$$\tau_{\mathrm{w}} = \mu \frac{\mathrm{d}U}{\mathrm{d}n} \,, \tag{26.11}$$

where μ denotes the dynamic viscosity of the fluid, U stands for the airspeed parallel to the surface, and n for the coordinate along the surface normal $\boldsymbol{n}$. This tangential fluid velocity gradient in the boundary layer is visualized qualitatively in Fig. 26.7.

The boundary layer will be laminar around the nose, with the fluid moving parallel to the surface. At some point (at a critical local Reynolds number), however, influenced by disturbances such as surface roughness, a transition to a turbulent boundary layer will occur: it is characterized by stochastic fluctuations, significantly thicker and producing substantially more friction than before the transition.

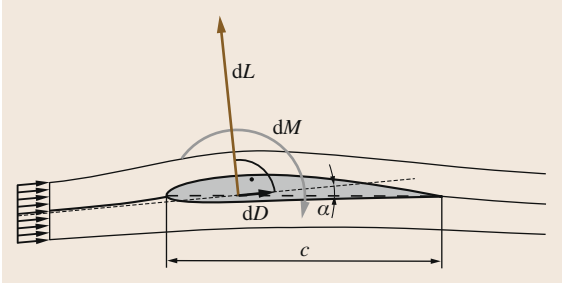


Fig. 26.8 Decomposition of the aerodynamic force by a 2-D flow around an airfoil: the section lift dL denotes the component perpendicular to the far-field inflow, and drag dD the one parallel to it

Avidbots Ex1016
Page 669 of 2154

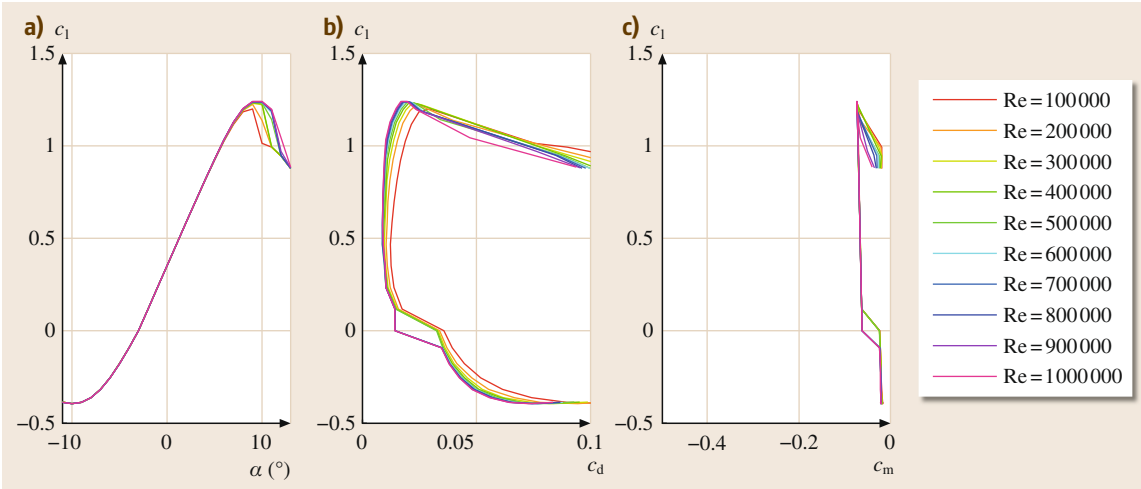


Fig. 26.9a–c SA7036 low-speed airfoil lift (**a**), drag (**b**), and moment (**c**) polars for various Reynolds numbers calculated by Javafoil

Section Lift, Drag, and Moment Representation with Dimensionless Coefficients

Historically and for practical reasons, the aerodynamic force is split into a component perpendicular to the inflow direction called *lift*, and a second one parallel to the inflow called *drag*. We write 2-D lift, drag, and moment as infinitesimal quantities $\mathrm{d}L$, $\mathrm{d}D$, and $\mathrm{d}M$, respectively, as opposed to L, D, and M designating physical forces of a whole airplane. Figure 26.8 visualizes these quantities. Furthermore, we define the angle of attack α as the angle between inflow direction and the chord line of length c connecting airfoil leading edge and trailing edge. Note that force and moment are reduced to the point at $0.25c$, i. e., one quarter of the chord behind the leading edge.

Dimension analysis suggests the formulation of aerodynamic forces and moments in terms of section lift, drag, and moment coefficients c_l, c_d, and c_m

$$\mathrm{d}L = \frac{1}{2}\rho V_\mathrm{t}^2 c_\mathrm{l} c\,\mathrm{d}y\,, \tag{26.12}$$

$$\mathrm{d}D = \frac{1}{2}\rho V_\mathrm{t}^2 c_\mathrm{d} c\,\mathrm{d}y\,, \tag{26.13}$$

$$\mathrm{d}M = \frac{1}{2}\rho V_\mathrm{t}^2 c_\mathrm{m} c^2\,\mathrm{d}y\,, \tag{26.14}$$

where V_t stands for the inflow speed and $\mathrm{d}y$ denotes an infinitesimal length element perpendicular to the 2-D flow (which can be interpreted as a length element into span-wise direction of an infinitely long wing).

These coefficients largely depend on the angle of attack α; but furthermore, the Reynolds and Mach numbers significantly influence them as well. The angle of attack dependences are typically given in the form of section lift, drag and moment polars, an example of which is provided in Fig. 26.9. Note that the drag component is originating both from viscous skin friction as well as form drag, caused by an asymmetric pressure distribution due to boundary layer development and separation. The lift curve shows its characteristic linear increase with increasing α for small angles of attack. The maximum and minimum lift values beyond which stall is entered are clearly visible in the lift polar. Note that the aerodynamic performance $c_\mathrm{l}/c_\mathrm{d}$ of the airfoil generally decreases with smaller Reynolds numbers as expected. The choice of reference point at $0.25c$ typically leads to a mostly constant moment coefficient when varying α, c_l, respectively, as can be seen in Fig. 26.9 as well.

Separation and Stall

At the upper side of the airfoil, the fluid is moving from the under-pressure region toward a higher pressure at the trailing edge; the slower moving fluid in the boundary layer will at some point not be able to follow this adverse pressure gradient, leading to flow separation. As the angle of attack is increased, the separation point suddenly moves far toward the leading edge: this condition is referred to as stall, with the catastrophic consequence of significant loss of lift and increase of drag. Figure 26.10 illustrates the changes in the flow and pressure distribution when varying the angle of attack. Note that the maximum lift and stall conditions are highly influenced by the choice of airfoil, Reynolds number, and Mach number.

26.3.3 Wing Aerodynamics

So far the 2-D flow characteristics around airfoils were treated. This will form the basis for the understanding and computation of lift and thrust forces generated on any type of aircraft; the following treatment of a finite wing serves as an important example of how to include three-dimensional (3-D) flow effects.

Recording lift and drag polars for a finite wing rather than just for its airfoil reveals less lift increase per angle of attack increase, less maximum lift, and higher drag at raised angles of attack. These observations are related to the concept of *induced flow* to be treated in the following.

Vortex System of a Wing

As a direct consequence of lift, we observe a downward flow deflection across an airfoil. This is intuitively explained with conservation of linear momentum as stated in (26.7). Assuming an inviscid and incompressible fluid, the flow may be modeled with potential field theory [26.13], where the velocity vector field is defined as the gradient of a scalar function. This concept allows for the insertion of singularities into a free stream, such as sources, sinks, and vortices. Figure 26.11 shows a first approximation using a single vortex – conceptually illustrating the flow characteristics around a simplified wing. The vortex system consists of the bound vortex and tip vortices; note that the vortex will in theory have to be closed to a ring by a starting vortex. In practice, i. e., in the presence of friction, the vortices will of course decay over time. Figure 26.12 illustrates the existence of tip vortices trailing an airplane wing.

The vortices induce a downwash area behind the wing; nevertheless, the trailing vortices will also induce some downward flow at the wing.

Induced Drag

With the simplified concept of the vortices around a wing in mind, we conclude that the wing lift induces downward flow, thus reducing the effective angle of attack when looking at the 2-D-flow of a wing cross-section. Figure 26.13 illustrates this reduction of the angle of attack from α_f (free stream) to α_e (effective) by the induced angle α_i that is caused by the induced flow component w_i. Note that this reduction of angle of attack is typically resulting in a smaller lift. Furthermore, when decomposing the lift into components parallel and perpendicular to the free stream velocity, it becomes apparent that a part dD_i of the lift results parallel to the effective inflow, thus will contribute to the overall drag of the wing. The integral of these components is referred to as induced drag. The actual amount of induced drag largely depends on the wing geometry; a variety of approaches have been employed in order to minimize induced drag, the most popular of which are winglets.

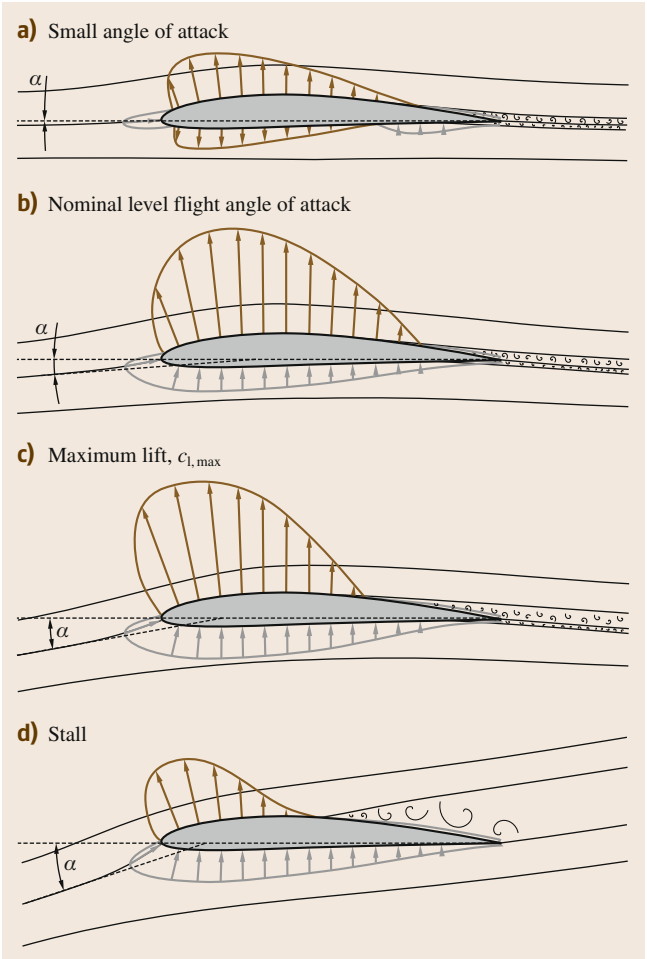


Fig.26.10a–d Changes in the flow characteristics with increasing angle of attack α. For a small α **(a)** the example nonsymmetric airfoil will generate some lift. **(b)** depicts nominal operation. At some α, the maximum lift $c_{l,max}$ is reached **(c)**. Beyond that angle of attack, stall occurs **(d)**

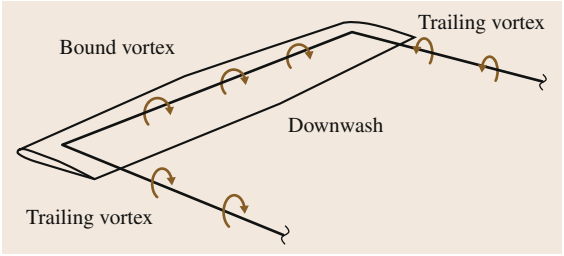


Fig. 26.11 Simplified representation of the wing vortex system: as a consequence of lift, a bound vortex is formed along with trailing wingtip vortices inducing downwash

Avidbots Ex1016
Page 671 of 2154



Fig. 26.12 Wake vortex study by NASA at Wallops Island: the tip vortices are visualized using colored smoke rising from the ground

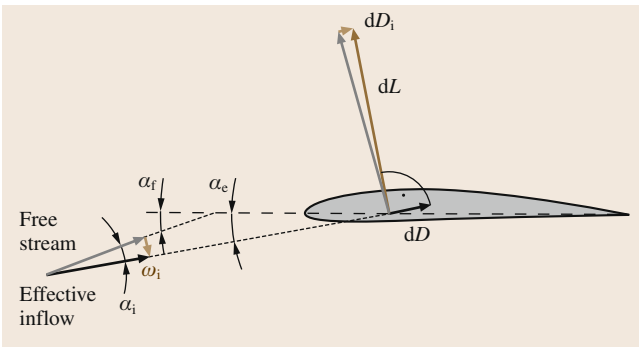


Fig. 26.13 Induced drag on a finite wing cross section: the induced downwash w_i causes a reduced effective angle of attack. As a consequence, the lift $\mathrm{d}L$ contains a component $\mathrm{d}D_\mathrm{i}$ parallel to the free stream velocity vector

For an approximately elliptical lift distribution, the induced drag coefficient can be roughly calculated as

$$c_{\mathrm{D,i}} = \frac{C_\mathrm{L}^2}{\pi e \Lambda} \,, \tag{26.15}$$

with the aspect ratio Λ and the Oswald efficiency e (deviation from the truly elliptic distribution) amounting from 0.7 to 0.85 for typical configuration.

Lifting Line Method

In the following, we present one example of how to numerically approximate the lift and drag distribution of a wing including induced drag. The lifting line method is a 2.5-D (two-and-a-half-dimensional) approach in which the induced flow is viewed as generated by several discrete horseshoe vortices rather than just one as introduced qualitatively earlier. Figure 26.14 depicts the geometry and variables involved. Note that the method only provides reliable results, if the assumption holds that spanwise flow is negligible; in particular, spanwise variation of parameters such as chord length and twist is supposed to be rather small. The Kutta–Joukowsky theorem relates circulation and lift; applied to a discrete wing segment, we obtain

$$\Gamma_k = \frac{1}{2} c_k c_{\mathrm{l},k}(\alpha_\mathrm{eff}) V_\mathrm{t} \,, \tag{26.16}$$

with the segment (index k) circulation Γ_k, the local chord length c_k, and the local airfoil lift coefficient $c_{\mathrm{l},k}$. The lift coefficient depends on the effective angle of attack: $\alpha_\mathrm{e} = \alpha_\mathrm{f} - \alpha_\mathrm{i}$. The induced downwash at position $\boldsymbol{m}_k$ is obtained by adding the induced speeds of all the individual vortices according to Biot–Savart

$$w_{\mathrm{i},k} = \sum_{j=1}^{n+1} \frac{\Gamma_j \boldsymbol{e}_\mathrm{V}}{4\pi \|(\boldsymbol{p}_j - \boldsymbol{m}_k) \times \boldsymbol{e}_\mathrm{V}\|} \times \left(1 + \frac{(\boldsymbol{p}_j - \boldsymbol{m}_k) \cdot \boldsymbol{e}_\mathrm{V}}{\|(\boldsymbol{p}_j - \boldsymbol{m}_k)\|}\right) \,, \tag{26.17}$$

where $\boldsymbol{e}_\mathrm{V}$ stands for the (unit) direction of flight. At $\boldsymbol{m}_k$, the induced angle of attack is calculated as

$$\alpha_{\mathrm{i},k} = \arctan \frac{w_{\mathrm{i},k}}{V_\mathrm{t}} \approx \frac{w_{\mathrm{i},k}}{V_\mathrm{t}} \,. \tag{26.18}$$

Together with the respective 2-D polar data, the above relations allow calculating the lift, drag, and moment distribution (with respect to the free inflow direction $\boldsymbol{e}_\mathrm{V}$) from a known circulation distribution, and can be summed and reduced to, e.g., the center of mass of a whole airplane. Note that the section lift coefficient is often approximated linearly as $c_\mathrm{l}(\alpha) \approx c_{\mathrm{l},0} + c_{\mathrm{l}_\alpha}\alpha$, which allows for a direct solution when applying (26.16) and (26.17). More accurate results, in particular in the domain near maximum lift, however, are obtained by using the nonlinear lift polar. In this case, a standard iterative numeric solver may be used. Furthermore, airfoil data with deflected control surfaces can be included, allowing to calculate control moments (and forces).

Note that the above method is just one example of numerically solving for the wing characteristics knowing the 2-D airfoil properties (polars) – well-suited for low-speed medium to high aspect ratio wings. For an overview on alternatives, the reader is referred to, e.g., [26.13].

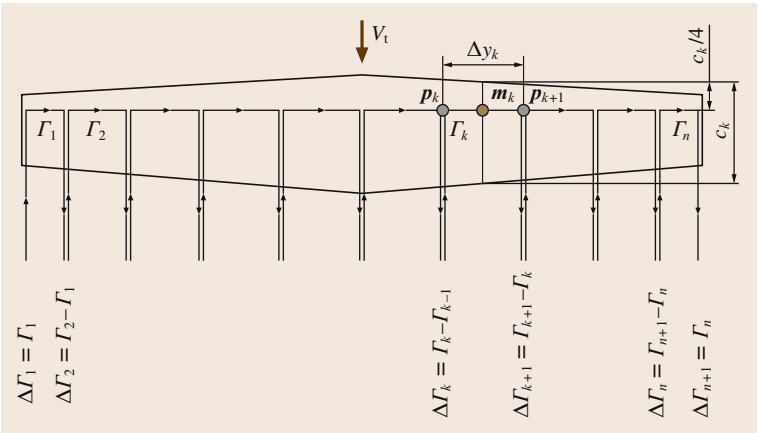


Fig. 26.14 k horse shoe vortices with circulation Γ_k placed on the wing to model the induced flow. The lifting line is imagined through the quarter-chord ($c_k/4$) locations. Vortex threads with strength $\Delta\Gamma_k$ are leaving the wing at the points $\boldsymbol{p}_k$ along the inflow and induce downwash at the locations $\boldsymbol{m}_k$

26.3.4 Performance of Rotors and Propellers

The propulsion mechanism found on many robotic aerial vehicles is commonly a specific configuration of propellers or rotors. In the case of a robotic airplane, forward facing propellers produce thrust forces compensating drag in forward flight. In case of a tail sitter or multicopter UAV, the propellers may be facing up (or down) and produce the main lift component compensating the vehicle's weight allowing it to hover in the air. Similarly, the more classic helicopter-type configurations (single rotor with tail rotor, coaxial rotor, tandem rotor, etc.), use rotors to generate the required thrust force to fly.

In order to decide on a suitable rotor or propeller geometry and to define requirements for the UAV motor drives, models must be available which allow for an assessment of the thrust and torque characteristics of a particular rotor or propeller. For this purpose, the blade element momentum theory (BEMT) has found widespread use, as it often provides a prediction accuracy which is acceptable for the UAV design process (despite its simplicity).

Blade Element Momentum Theory

One of the basic difficulties in aerodynamic rotor and propeller studies is the prediction of the induced inflow velocities discussed in Sect. 26.3.3. BEMT addresses this problem by combining two simple modeling approaches, namely momentum theory (MT) and blade element theory (BET) which individually cannot directly resolve this issue in an accurate manner [26.14].

The basic idea of momentum theory is to consider the revolving propeller or rotor as a propulsion disk which produces a thrust force by accelerating the surrounding (incompressible) air mass passing through it. A boundary volume is defined encapsulating the propulsion disc. Subsequently, the laws of mass, momentum and power conservation are formulated across the boundaries of the defined control volume. The concept of this propulsion disk as well as the corresponding control volume are visualized in Fig. 26.15.

From this simplistic model, two main conclusions may be drawn. First and foremost, it is possible to establish a relation between the induced velocity v_i at the propulsion disk and the produced thrust force T. In normalized form, it can be expressed as

$$c_T = 2\lambda_i(\lambda_i + \lambda_\infty) \,. \tag{26.19}$$

To simplify notation, the external airflow velocity v_∞ as well as the induced velocity v_i have been normalized with the rotor or propeller tip speed ΩR

$$\lambda_i = \frac{v_i}{\Omega R}, \qquad \lambda_\infty = \frac{v_\infty}{\Omega R} \,, \tag{26.20}$$

and the nondimensional thrust coefficient c_T is defined as

$$c_T = \frac{T}{\rho(\pi R^2)(\Omega R)^2} \,. \tag{26.21}$$

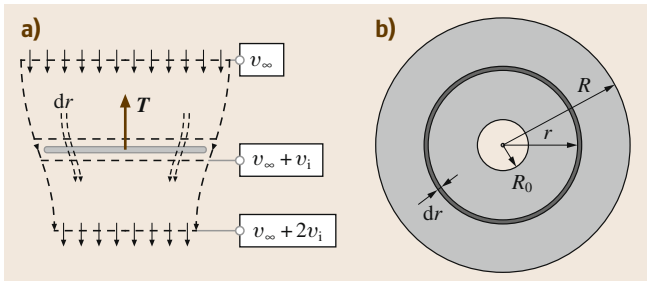


Fig. 26.15 **(a)** Side view on the slipstream control volume encompassing the MT propulsion disc. **(b)** Top view on MT propulsion disk with incremental annular section and root cutout

The parameter ρ corresponds to the density of air, R to the rotor or propeller radius, and Ω to the rotor or propeller angular speed.

In case of a robotic airplane, the velocity v_∞ at the start of the control volume corresponds to the body forward flight velocity V_t and in case of a rotorcraft configuration to the body climb respectively descent rate w.

Similarly, an incremental expression for the thrust coefficient may be found from MT by evaluating the mass, moment, and power conservation laws over an annular ring of the defined control volume only. The corresponding expression can be found as

$$\mathrm{d}C_\mathrm{T}^\mathrm{MT} = 4\lambda_\mathrm{i}(\lambda_\mathrm{i} + \lambda_\infty)\bar{r}\,\mathrm{d}\bar{r}\,, \tag{26.22}$$

where $\bar{r} = \frac{r}{R}$ and $\mathrm{d}\bar{r} = \frac{\mathrm{d}r}{R}$ are the normalized radial location and the radial increment of the propulsion disk annulus.

Another relevant conclusion that may be drawn from MT is that ideally, the induced component v_i of the slipstream velocity at the propulsion disk will accelerate to two times its initial value before leaving the control volume. In consequence to this acceleration of the flow field, the radial slipstream boundary will (in the ideal case) contract to half the propulsion disk area at the end of the control volume.

For the BET approach, the modeling process starts by investigating the aerodynamic lift and drag forces $\mathrm{d}L$ and $\mathrm{d}D$ on an individual rotor or propeller blade revolving around its shaft. These lift and drag forces produced by each airfoil segment depicted in Fig. 26.16, contribute to the total thrust and torque increments $\mathrm{d}T$ and $\mathrm{d}Q$ of the respective rotor or propeller annular section. The corresponding relation can be established as

$$\mathrm{d}T = N_\mathrm{b}(\mathrm{d}L - \Phi\,\mathrm{d}D) \approx N_\mathrm{b}\,\mathrm{d}L\,, \tag{26.23}$$

$$\mathrm{d}Q = N_\mathrm{b} r(\mathrm{d}D + \Phi\,\mathrm{d}L)\,. \tag{26.24}$$

In this context, N_b represents the number of rotor or propeller blades and Φ corresponds to the local inflow angle which is assumed to remain small. Under this assumption, the inflow angle Φ can be directly derived as the ratio between the local perpendicular inflow velocity $U_\mathrm{P} \approx v_\mathrm{i} + v_\infty$ and the tangential velocity $U_\mathrm{T} \approx \Omega r$ visualized in Fig. 26.16b. Additionally, the assumption introduced in (26.23) is justified by the fact that at low angles of attack α, the drag forces $\mathrm{d}D$ are at least one order of magnitude smaller than the corresponding lift forces $\mathrm{d}L$.

Based on (26.23) and the definition of the lift increment (26.12), the local thrust coefficient at each radial blade station r may be derived as

$$\mathrm{d}C_\mathrm{T}^\mathrm{BET} = \frac{1}{2}\sigma c_\mathrm{l}\bar{r}^2\mathrm{d}\bar{r}, \quad \text{with} \quad \sigma = \frac{N_\mathrm{b}c}{\pi R}\,. \tag{26.25}$$

The parameter c corresponds to the local blade chord and σ is the so-called rotor or propeller *solidity*. The solidity is a rough metric representing how much of a propulsion disk is covered by rotor or propeller blades. The aerodynamic parameter $c_\mathrm{l} = c_\mathrm{l}(\alpha, \mathrm{Re}, \mathrm{Ma})$ corresponds to the airfoil lift coefficient in function of the local angle of attack α and the local Reynolds and Mach numbers Re and Ma.

Accordingly, as established in (26.25), the thrust produced by a rotor or propeller strongly depends on the angle of attack α, which itself is a function of the local airfoil pitch angle θ and the inflow angle Φ

$$\alpha = \theta - \Phi \approx \theta - \frac{U_\mathrm{P}}{U_\mathrm{T}}\,. \tag{26.26}$$

In conclusion, MT as well as BET are capable of establishing a meaningful relation between the induced velocities v_i and the resulting thrust force T. However, none of the two theories are capable of accurately predicting rotor or propeller performance as either the radial distribution of the induced velocity or the radial distribution of the thrust coefficient must be known to compute the other.

The basic idea behind BEMT is to combine the thrust expression (26.22) resulting from MT with the thrust expression (26.25) from BET to compute the induced inflow velocity independently of the thrust force. Different BEMT implementations are possible depending on how willing one may be to introduce further assumptions for the section lift coefficient c_l.

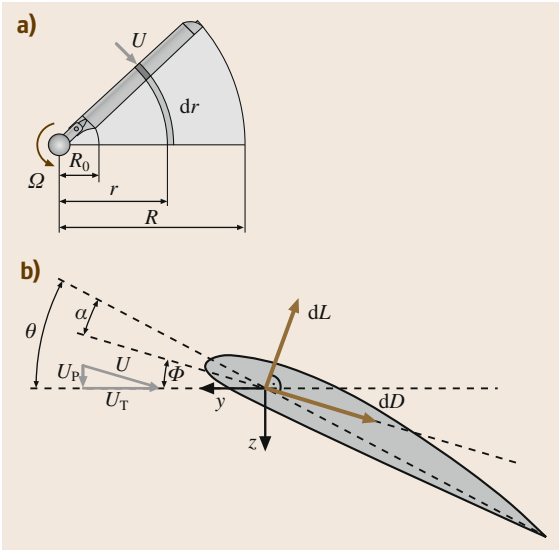


Fig. 26.16 **(a)** Rotor blade revolving around its shaft. **(b)** Blade element of a revolving rotor blade

Reference [26.14] presents a straightforward approach tailored toward helicopter rotors operating bellow stall by introducing a linear model for c_l in function of the angle of attack

$$c_l = c_{l1}\alpha + c_{l0} \,. \tag{26.27}$$

The parameters c_{l0} and c_{l1} can be computed from the lift polars of a particular airfoil geometry for a given range of angles of attack, Reynolds, and Mach numbers. This linear approximation may have limited validity for very low Reynolds numbers and strongly cambered airfoils but is in general acceptable for many typical airfoil geometries found on UAV rotors and propellers. Assembling (26.22) and (26.25) under the assumption (26.27), an algebraic expression of the radial induced inflow distribution can be derived as

$$\lambda_i(r, \lambda_\infty) = \sqrt{A^2 + B} - \overline{A} \,, \tag{26.28}$$

$$A = \frac{\sigma c_{l1}}{16} - \frac{\lambda_\infty}{2} \,,$$
$$\overline{A} = \frac{\sigma c_{l1}}{16} + \frac{\lambda_\infty}{2} \,,$$
$$B = \frac{\sigma c_{l1}}{8}\theta' \overline{r} \,.$$

Note that the lift-curve offset c_{l0} has been absorbed in the virtual pitch angle

$$\theta' = \theta + \frac{c_{l0}}{c_{l1}} \,, \tag{26.29}$$

to simplify the notation.

Once the approximate radial distribution of inflow velocities has been found, the local rotor or propeller thrust increments (26.25) can be computed. Similarly, the rotor or propeller torque increments derived from BET as

$$\mathrm{d}c_Q^{\mathrm{BET}} = \mathrm{d}c_{Qi} + \mathrm{d}c_{Q0} \,, \tag{26.30}$$

$$\mathrm{d}c_{Qi} = \frac{1}{2}\sigma c_l(\lambda_i + \lambda_\infty)\overline{r}^3 \,\mathrm{d}\overline{r} \,, \tag{26.31}$$

$$\mathrm{d}c_{Q0} = \frac{1}{2}\sigma c_d \overline{r}^3 \,\mathrm{d}\overline{r} \,, \tag{26.32}$$

can be evaluated based on the inflow distribution given in (26.28). For clarity, the total torque coefficient increment has been separated into its induced component $\mathrm{d}c_{Qi}$ originating from the lift forces and its profile component $\mathrm{d}c_{Q0}$ due the drag forces. The aerodynamic drag coefficient $c_d = c_d(\alpha, \mathrm{Re}, \mathrm{Ma})$ can be approximated using a quadratic function in dependency of the angle of attack [26.14]

$$c_d = c_{d2}\alpha^2 + c_{d1}\alpha + c_{d0} \,. \tag{26.33}$$

The parameters c_{d0}, c_{d1}, and c_{d2} can be computed from the drag polars of the modeled airfoil.

Consequently, the thrust and torque increments may be integrated along the radial direction of the rotor or propeller disk to compute the total thrust and torque coefficients

$$c_T = \int_0^R \mathrm{d}c_T^{\mathrm{BET}} \,, \tag{26.34}$$

$$c_Q = \int_0^R \mathrm{d}c_Q^{\mathrm{BET}} \,. \tag{26.35}$$

These thrust and torque integrals are usually evaluated numerically, as the blade pitch $\theta = \theta(r)$ as well as the blade chord $c = c(r)$ may be nonlinear functions of the radial direction r (blade *twist* and *taper*).

Finally, note that the presented theory may be extended to provide performance estimates under lateral inflow velocities such as in case of a rotorcraft in forward flight and may also be used to asses other types of rotor or propeller configurations such as, e.g., the coaxial rotor. Also note that the prediction accuracy of BEMT tools can be further improved by accounting for tip-loss effects and the nonlift producing rotor or propeller hub, e.g., using the Prandtl tip-loss function also presented in [26.14] and incorporating a root cutout radius R_0.

The resulting predictions are generally in good agreement with experimental data – nevertheless, an experimental verification is strongly recommended.

26.3.5 Drag

The sources of drag on aircraft are manifold: historically, the distinction between the lift-dependent *induced drag* and *parasite drag* is made. The latter is further subdivided into *skin friction drag* due to viscous shear stress at the surface, and *form drag*, generated by pressure loss along bodies (i. e., due to boundary layer development or even flow separation). Both these components contribute to the airfoil *profile drag*, i. e., the section drag coefficient c_d introduced before.

On a whole aircraft, many more drag sources are distinguished. For an exhaustive overview, the interested reader is referred to [26.13]. In the following, we will present an overview of drag generated by different typical shapes; note that when simply summing drag of different shapes associated with aircraft parts, the result may be a helpful initial estimate, but can be inaccurate, because of neglecting the interaction of flows resulting in *interference drag*. Depending on the stage of the design process or the desired modeling accuracy, 2.5-D

Avidbots Ex1016
Page 675 of 2154

computations or even full 3-D computational fluid dynamics (CFD) simulations might be necessary to satisfy the needs of aerodynamics calculations.

Skin Friction

The simple but important example of a flat plate of length l in parallel flow is well studied. As introduced in the airfoil theory Sect. 26.3.2, the boundary that develops will be laminar near the leading edge and transitions into a turbulent one, generating more drag, at some point downstream. The friction coefficient is defined as

$$c_\mathrm{f} = \frac{2D_\mathrm{f}}{\rho V_\mathrm{t}^2 S_\mathrm{w}} , \tag{26.36}$$

with the wetted surface S_w and the friction drag force D_f. According to [26.13], the coefficients can be approximated by

$$\text{Laminar}: \quad c_\mathrm{f} = 1.328\mathrm{Re}_\mathrm{l}^{-0.5} , \tag{26.37}$$

$$\text{Turbulent}: \quad c_\mathrm{f} = 0.455(\log_{10}\mathrm{Re}_\mathrm{l})^{-2.58} . \tag{26.38}$$

Note that the point of transition is depending on the local Reynolds number $\mathrm{Re}_x = \rho V_\mathrm{t} x/\mu$, where x denotes the coordinate along the flow from the leading edge of the plate. Depending on the surface roughness and ambient turbulence, the critical (transition) Reynolds number varies; as an average guess for a flat plate, it will be in the order of $\mathrm{Re}_{x,\mathrm{crit}} = 3\times10^5$.

Drag Coefficients for Selected Bodies

In the following, drag coefficients for a selection of 2-D and 3-D bodies of rotation are given, obtained from [26.13]. Table 26.3 overviews a category of bodies, the drag coefficients of which are largely indifferent to the Reynolds number, owing to their geometrically defined (sharp edge) flow separation point.

Rounder bodies, most prominently the cylinder in cross-flow or the sphere, however, show a distinctively different behavior quantified in Table 26.4: below a critical Reynolds number $\mathrm{Re}_\mathrm{crit} \approx 4\times10^5$, the drag coefficient is significantly higher, where separation occurs before boundary layer transition. In contrast, above the critical Reynolds number, the turbulent, more energetic boundary layer separates only further downstream, reducing the wake and thus the amount of form drag.

A third important object category is formed by streamlined and fuselage-like bodies: due to their comparably high skin friction part, the fineness ratio largely influences the drag coefficient (along with the Reynolds number). The fineness is defined as body length divided by body diameter. We introduce a volumetric drag coefficient as $c_\mathrm{Dm} = 2D/(\rho V_\mathrm{t}^2 V_\mathrm{m}^{2/3})$ with the body volume V_m. Interestingly, this is minimal and approximately constant at fineness ratios between 4 and 10, which provides a range for optimal sizing of such a body when a certain volume needs to be fitted. The values are given in Table 26.5.

Table 26.3 Bodies the drag coefficients of which are largely Reynolds number independent

	c_D (2-D)		c_D (3-D)	
⟶		1.98		1.18
⟶		2.0		1.0–1.2
⟶		1.3		0.7
⟶		2.0		1.1
⟶		2.2		1.7
⟶		1.4		0.4

26.3.6 Aircraft Dynamics and Flight Performance Analysis

Sections 26.3.1–26.3.5 shortly presented the basic theory of wings, rotors, and propellers as well as a few tools to assess their respective aerodynamic performance. To develop fully functional UAV platforms, merely evaluating the individual flight mechanisms is a good starting point but generally not sufficient.

Designing high-performance aircraft systems requires a fundamental understanding of how the respective design parameters affect the full flight dynamic response and application-specific capabilities. In order to comprehend how design changes affect an aerial

Table 26.4 Sphere and cylinder drag coefficients

	c_D (2-D)	c_D (3-D)
⟶		
$\mathrm{Re} < \mathrm{Re}_\mathrm{crit}$	1.1	0.4
$\mathrm{Re} > \mathrm{Re}_\mathrm{crit}$	0.27	0.15

Table 26.5 Volumetric drag coefficients for fineness 4–10 (largely turbulent boundary layer)

Fuselages and nacelles	$c_\mathrm{Dm} \approx 0.027$
Streamlined bodies	$c_\mathrm{Dm} \approx 0.024$

Avidbots Ex1016
Page 676 of 2154

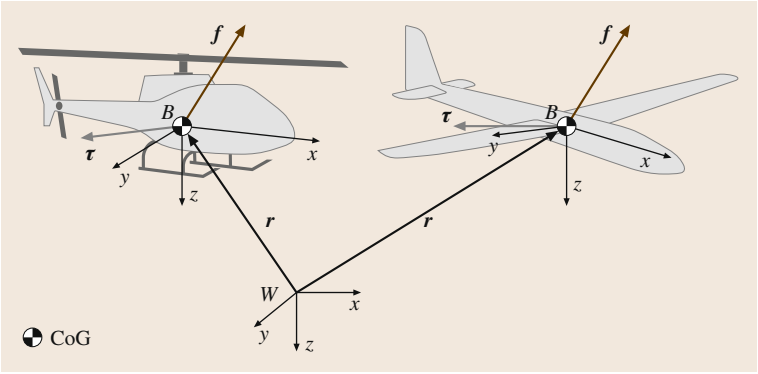


Fig. 26.17 Coordinate frames with external forces and moments of rotorcraft and fixed-wing UAVs

robotic system, representative models of its flight dynamics are required. Such models must be capable of capturing the dominant system dynamics within the relevant part of the flight envelope. Furthermore, especially in an interdisciplinary field such as robotics, these models must be accessible to the nonaerodynamic expert (the roboticist) and thus need to be simple enough to provide the required insight for the aircraft design process.

As many other types of robots, robotic flight platforms may be treated as a multibody system where a set of interlinked bodies exchanges kinetic and potential energy under the influence of external forces and moments. For aircraft systems it is common to treat the entire aircraft as a single rigid body first, with related body coordinate frame attached, as visualized in Fig. 26.17. Additional dynamics such as for example rotor flapping (as in case of a helicopter system) may be appended to these body dynamics in a subsequent step.

The modeling process thus starts by treating the aircraft system as a rigid body affected by external forces $\boldsymbol{F}$ and external moments $\boldsymbol{\tau}$. Using the Newton–Euler formalism to derive the aircraft body dynamics, one can directly write down the linear and angular momentum balance for a single rigid body

$$\begin{aligned} m({}^B\dot{\boldsymbol{v}} + {}^B\boldsymbol{\omega} \times {}^B\boldsymbol{v}) &= {}^B\boldsymbol{F} \,, \\ {}^B\mathbf{I}\,{}^B\dot{\boldsymbol{\omega}} + {}^B\boldsymbol{\omega} \times ({}^B\mathbf{I}\,{}^B\boldsymbol{\omega}) &= {}^B\boldsymbol{\tau} \,. \end{aligned} \tag{26.39}$$

For simplicity's sake, (26.39) is usually expressed with respect to a body fixed frame B located in the center of gravity of the aircraft. The velocity vectors ${}^B\boldsymbol{v} = (u, v, w)^{\mathrm{T}}$ and ${}^B\boldsymbol{\omega} = (p, q, r)^{\mathrm{T}}$ thus represent the aircraft linear and angular velocities with respect to B. The inertial properties of the above body dynamics are defined by the aircraft's total mass m and its second mass moment of inertia ${}^B\mathbf{I}$ also expressed with respect to B and its origin.

The most relevant contribution to the forces ${}^B\boldsymbol{F}$ and the moments ${}^B\boldsymbol{\tau}$ originates from the aerodynamic flight components such as wings, propellers and rotors. By integrating (26.39) over time one may compute a prediction of the aircraft's dynamic response to these external forces and moments and thus the evolution of its absolute pose. This pose is commonly represented by the position of the vehicle's center of gravity ${}^W\boldsymbol{r}_B = (x, y, z)^{\mathrm{T}}$ as well as the vehicle's orientation relative to an earth fixed world frame W which is considered inertial. The aircraft orientation is commonly represented using rotation matrices or quaternions. In the case of the rotation matrix representation, the aircraft orientation may be parameterized in three dimensional space by three consecutive rotations with the roll, pitch, and yaw angles $\varphi \in [-\pi, \pi]$, $\theta \in [-\pi/2, \pi/2]$ and $\psi \in [-\pi, \pi]$ as

$${}^W\boldsymbol{R}_B = \boldsymbol{R}_Z(\psi)\boldsymbol{R}_Y(\theta)\boldsymbol{R}_X(\varphi) \,. \tag{26.40}$$

In this case, relations between the body frame velocities ${}^B\boldsymbol{v}$ and ${}^B\boldsymbol{\omega}$ and the world frame pose can be expressed as

$$\begin{aligned} {}^W\dot{\boldsymbol{r}} &= {}^W\boldsymbol{R}_B\,{}^B\boldsymbol{v} \,, \\ {}^W\dot{\boldsymbol{R}}_B &= {}^W\boldsymbol{R}_B[{}^B\boldsymbol{\omega}]^{\times} \,, \end{aligned} \tag{26.41}$$

where $[{}^B\boldsymbol{\omega}]^{\times}$ corresponds to the skew-symmetric matrix of the vector ${}^B\boldsymbol{\omega}$.

In a minimal form, the orientation dynamics can be expressed in terms of roll, pitch, and yaw angles

$$\begin{pmatrix} \dot{\varphi} \\ \dot{\theta} \\ \dot{\psi} \end{pmatrix} = \underbrace{\begin{pmatrix} 1 & \sin\varphi\tan\theta & \cos\varphi\tan\theta \\ 0 & \cos\varphi & -\sin\varphi \\ 0 & \sin\varphi/\cos\theta & \cos\varphi/\cos\theta \end{pmatrix}}_{\mathbf{J}_r} {}^B\boldsymbol{\omega} \,. \tag{26.42}$$

Note that the Jacobian $\mathbf{J}_r$ becomes singular at the boundaries of $\theta = \pm\pi/2$.

As a singularity-free, but still compact representation of orientation, quaternions may be used. Using the representation ${}^W\boldsymbol{q}_B = (q_w, q_x, q_y, q_z)^{\mathrm{T}}$ with real part q_w, the rotational kinematics become

$$ {}^W\dot{\boldsymbol{q}}_B = \frac{1}{2}\boldsymbol{\Omega}({}^B\boldsymbol{\omega})\,{}^W\boldsymbol{q}_B \,, \tag{26.43} $$

with the matrix $\boldsymbol{\Omega}$ defined as

$$ \boldsymbol{\Omega}({}^B\boldsymbol{\omega}) = \begin{pmatrix} 0 & {}^B\boldsymbol{\omega}^{\mathrm{T}} \\ -{}^B\boldsymbol{\omega} & [{}^B\boldsymbol{\omega}]^{\times} \end{pmatrix} . \tag{26.44} $$

In summary (using quaternions), we thus have the equations of motion

$$ \begin{aligned} {}^W\dot{\boldsymbol{r}} &= {}^W\boldsymbol{R}_B\,{}^B\boldsymbol{v} \,, \\ {}^W\dot{\boldsymbol{q}}_B &= \frac{1}{2}\boldsymbol{\Omega}({}^B\boldsymbol{\omega})\,{}^W\boldsymbol{q}_B \,, \\ {}^B\dot{\boldsymbol{v}} &= \frac{1}{m}{}^B\boldsymbol{F} - {}^B\boldsymbol{\omega}\times{}^B\boldsymbol{v} \,, \\ {}^B\dot{\boldsymbol{\omega}} &= {}^B\mathbf{I}^{-1}[{}^B\boldsymbol{\tau} - {}^B\boldsymbol{\omega}\times({}^B\mathbf{I}\,{}^B\boldsymbol{\omega})] \,. \end{aligned} \tag{26.45} $$

Note that the external forces and moments are related to the system's actuator inputs $\boldsymbol{u}$, the vehicle's orientation with respect to W and they typically are also functions of the linear and angular body motion as well as additional dynamics terms, represented here by the vector $\boldsymbol{\varepsilon}_r$

$$ {}^B\boldsymbol{F} = {}^B\boldsymbol{F}({}^W\boldsymbol{r}, {}^W\boldsymbol{q}_B, {}^B\boldsymbol{v}, {}^B\boldsymbol{\omega}, \boldsymbol{\varepsilon}_r, \boldsymbol{u}) \,, \tag{26.46} $$

$$ {}^B\boldsymbol{\tau} = {}^B\boldsymbol{\tau}({}^W\boldsymbol{r}, {}^W\boldsymbol{q}_B, {}^B\boldsymbol{v}, {}^B\boldsymbol{\omega}, \boldsymbol{\varepsilon}_r, \boldsymbol{u}) \,. \tag{26.47} $$

The additional dynamics $\boldsymbol{\varepsilon}_r$ may account for structural dynamics such as rotor flapping in case of a robotic helicopter or relevant actuator dynamics.

The resulting nonlinear system dynamics can usually be cast into state-space form and represented as

$$ \begin{aligned} \dot{\boldsymbol{x}} &= \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}) \,, \\ \boldsymbol{x} &= (\boldsymbol{x}_b, \boldsymbol{x}_r)^{\mathrm{T}} \,, \\ \boldsymbol{u} &= (u_1, \ldots, u_N)^{\mathrm{T}} \,, \end{aligned} \tag{26.48} $$

where the nonlinear functions $\boldsymbol{f}$ define the rate of change of the aircraft body states $\boldsymbol{x}_b$ as well as the additional states $\boldsymbol{x}_r$ affected by the set of N actuator inputs u_1 to u_N.

In order to gain a deeper understanding of how (26.48) is affected by changes of the flight system's geometric, structural, inertial, and aerodynamic parameters, three main problems are commonly of relevance [26.15]:

- The *trim problem* deals with the computation of the set of actuator inputs $\boldsymbol{u} = \boldsymbol{u}_0$ under which the nonlinear dynamic system presented in (26.48) remains in a desired trim point $\boldsymbol{x} = \boldsymbol{x}_0$ and thus $\boldsymbol{f}(\boldsymbol{x}_0, \boldsymbol{u}_0) = \boldsymbol{0}$. The most simple example of such a trim point $\boldsymbol{x}_0$ is the hover condition for a rotorcraft system where one may want to find the required rotor speed Ω_0 to hover or the steady forward flight condition for a fixed-wing UAV at a forward velocity V_t.
- The topic of *stability* deals with the question of how easily the system (26.48) will deteriorate from a specific trim condition $(\boldsymbol{x}_0, \boldsymbol{u}_0)$ under the influence of small disturbances $\Delta\boldsymbol{x}$ and $\Delta\boldsymbol{u}$. This investigation commonly involves the linearization of (26.48) according to

$$ \begin{aligned} \boldsymbol{A} &= \left(\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\right)_{\boldsymbol{x}=\boldsymbol{x}_0, \boldsymbol{u}=\boldsymbol{u}_0} \,, \\ \boldsymbol{B} &= \left(\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}}\right)_{\boldsymbol{x}=\boldsymbol{x}_0, \boldsymbol{u}=\boldsymbol{u}_0} \,, \\ \Delta\dot{\boldsymbol{x}} &= \boldsymbol{A}\Delta\boldsymbol{x} + \boldsymbol{B}\Delta\boldsymbol{u} \,, \end{aligned} \tag{26.49} $$

  where the eigenvalues and vectors of $\boldsymbol{A}$ will provide deeper insight into the motion characteristics and stability properties of an aircraft.
- Analyzing the *System Response*

$$ \boldsymbol{x}(t) = \boldsymbol{x}(0) + \int_0^t \dot{\boldsymbol{x}}\,\mathrm{d}t \tag{26.50} $$

  to characteristic inputs such as steps, pulses or specific input frequencies will provide additional information about the flight characteristics of a specific aircraft configuration.

These modeling and analysis concepts commonly find wide applicability for various types of robotic flight configurations and will be discussed in more detail for the specific UAV types presented hereafter.

Actuator Dynamics

Deriving the aerodynamic forces is combined with the dynamic equations of motion in order to assemble a complete model of the flying vehicle. However, as the employed actuators are of naturally limited bandwidth, accurate modeling furthermore requires the integration of the relevant motor or servo dynamics.

Nowadays, motors utilized in small size unmanned systems often belong to the category of brushless direct-current (DC) electric motors (BLDC). BLDCs are synchronous motors powered by a DC electric source via an integrated switching power supply. The equations of motion for a such a system are essentially nonlinear and

rather complex. However, working with small UAS, we may solely focus on the input–output dynamics which can be described with the following transfer function

$$\frac{\Delta\omega(s)}{\Delta Q_\mathrm{m}(s)} = \frac{-K_\mathrm{m}(1+\tau_\alpha s)}{(1+\tau_\mathrm{m}s)(1+\tau_\alpha s)+K_\mathrm{m}K_\alpha(Ki_{e0})^2}\,, \tag{26.51}$$

where $\Delta\omega(s)$, $\Delta Q_\mathrm{m}(s)$ correspond to the Laplace expressions of the linearized angular velocity and input torque, K_m is the mechanical gain, τ_m represents the mechanical time constant, K_α is the rotor gain, τ_α is the rotor time constant, K depends on electromagnetic properties of the motor and i_{e0} denotes the stator current linearization point [26.16]. Often, a satisfactory speed controller and BLDC dynamics description is obtained as the relation between a reference angular velocity and the actual output taking the even simpler first-order form

$$\frac{\Delta\omega(s)}{\Delta\omega^r(s)} = \frac{1}{1+\tau_\mathrm{mc}s}\,, \tag{26.52}$$

with the time constant τ_mc of the controlled motor.

Accounting for motor dynamics is essential for high-bandwidth control of agile vehicles that highly depend on such actuators (i. e., multirotors). However, in several other UAS configurations such as fixed-wing vehicles or conventional helicopters may, if needed, rather account for servo dynamics acting on control surfaces or a swashplate. Again, the relevant servo angle dynamics can be captured by an identified first-order transfer function of the form $1/(1+\tau_\mathrm{s}s)$, with the servo time constant τ_s.

26.4 Airplane Modeling and Design

Ever since the beginning of aviation, a broad spectrum of airplanes has been built and operated successfully: size, speed, and maneuverability vary widely and as a function of application. Since design and modeling are strongly related, we want to first give an overview of the physical principles common to all such configurations, and provide analysis tools for characterizing static and dynamic properties of an airplane. The design problem somewhat constitutes the inverse problem: for specified target characteristics, the engineer needs to find a suitable configuration; we therefore provide a summary of design guidelines aimed at fast convergence to a suitable design. Finally, a simple and classical autopilot scheme is presented underlining the need for models also at that stage.

26.4.1 Forces and Moments

Consider Fig. 26.18 for the introduction of airplane geometry definitions and main forces. Forces and moments are reduced to the airplane center of gravity (COG). Note that the angle of attack (AOA) α is defined as the angle between the x-axis and the true airspeed vector $\boldsymbol{v}_\mathrm{t}$ projected into the body x–z-plane, β denotes the sideslip angle, causing a typically unwanted sideslip force Y, L, and D denote lift and drag, W stands for the weight and T for thrust, which may act into a direction different from x (at a thrust angle ϵ_T). We furthermore write the aerodynamic moment vector as ${}^B\boldsymbol{\tau}_\mathrm{A} = [L_\mathrm{A}, M_\mathrm{A}, N_\mathrm{A}]^\mathrm{T}$. Also note the introduction of the main control surfaces that are designed to mainly influence the aerodynamic moment: with ailerons, elevator, and rudder, the roll (L_A), pitch (M_A), and yaw (N_A) moments are controlled. The indicated flaps, if available, are used for increasing lift for take-off and landing, in order to achieve a slower minimum speed.

Aerodynamic Forces and Moments

The aerodynamic forces and moments can be modeled to various accuracy using full 3-D CFD or with 2.5-D tools: Sect. 26.3.3 overviews such an approach which can be used to model the aerodynamic surfaces in incompressible flow. For enhanced accuracy, fuselages may be

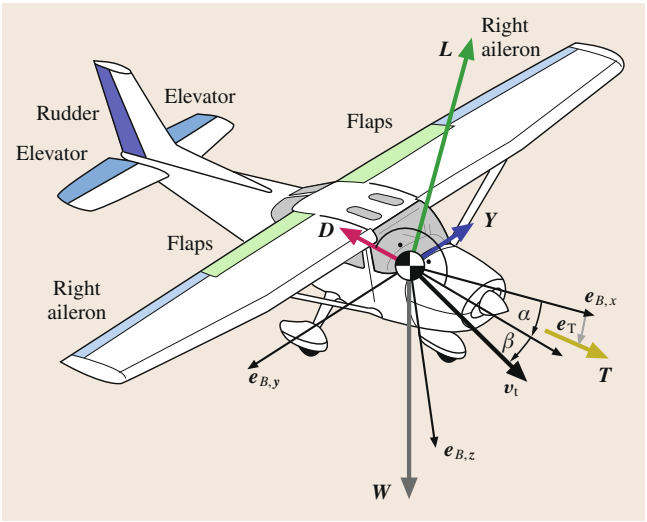


Fig. 26.18 Geometric definitions and main forces acting on the airplane (general case, not in equilibrium)

considered using again a combination of potential flow (placing singularities) and boundary layer theory. Respective ready-to-use software such as AVL [26.17] and XFLR [26.12] is available for free.

The forces and moments may again be written with dimensionless coefficients as

$$L = \frac{1}{2}\rho V_t^2 c_L A \,, \tag{26.53}$$

$$D = \frac{1}{2}\rho V_t^2 c_D A \,, \tag{26.54}$$

$$M_A = \frac{1}{2}\rho V_t^2 c_M \bar{c} A \,, \tag{26.55}$$

with the wing area A, the mean chord length $\bar{c}$, and the true airspeed $V_t = \|\boldsymbol{v}_t\|$. The moments L_A and N_A are made dimension-less with the wingspan b rather than the chord length.

Static Performance Considerations

Having characterized lift and drag of an airplane, three operating points are of particular interest.

First, stall is occurring at $c_{L,max}$. This condition can be directly translated into constant-speed level-flight stall speed by applying the lift balance $L = mg$.

Second, the maximum c_L/c_D ratio, or the *glide ratio* characterizes the maximum aerodynamic efficiency, i. e., the operating point for maximum range (assuming constant propulsive efficiency).

Finally, the maximum c_L^3/c_D^2 ratio, or the *climb factor* describes the condition at which power consumption is minimized, thus maximizing flight time (again assuming constant propulsion unit efficiency).

The latter two conditions have direct interpretation in gliding (or propulsion shut-off), in terms of maximum distance reached per altitude lost and minimum sink rate, respectively. Again, corresponding velocities can be found using the lift balance.

Thrust

For detailed insight into the variety of propulsion systems and respective models, the interested reader is referred to [26.13, 18]. As an approximation for the important case of a propeller, the BEMT method as described in Sect. 26.3.4 is suggested. For many applications, choosing the propeller speed as the system input and neglecting motor dynamics is sufficient.

26.4.2 Static Stability

Various forms of stability constitute central characteristics of an airplane related to whether or not it can be flown by a human pilot or flight controller. Simple stability criteria can be derived by requiring reaction forces and moments to be opposing a disturbance. We assume stationary conditions in the sense of constant linear and angular speeds: the respective force and moment balance is typically straightforward to apply in order to determine the starting point of the stability analysis.

Longitudinal Static Stability

We will take a close look at the example of longitudinal static stability playing a central role in airplane analysis and design. Leaving aside possible influence of the propulsion unit, the respective directional stability criterion is stated as

$$\frac{\partial c_M}{\partial \alpha} < 0 \,, \tag{26.56}$$

at the equilibrium condition $c_M = 0$. Figure 26.19 illustrates an exemplary moment coefficient as a function of AOA. Note that elevator actuation will *move* this curve up and down, and with it the equilibrium point (2) toward higher or lower angles of attack (i. e., lower or higher trimmed speeds). Figure 26.20 illustrates the forces and moments in the stable equilibrium

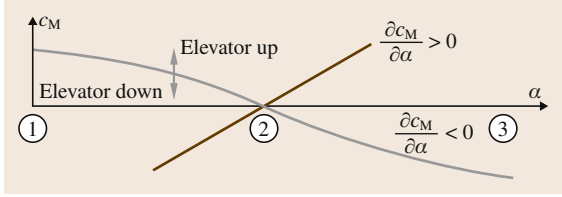


Fig. 26.19 Moment coefficient c_M as a function of AOA α the equilibrium point (2) is stable for $\partial c_M/\partial\alpha < 0$ (*brown curve*)

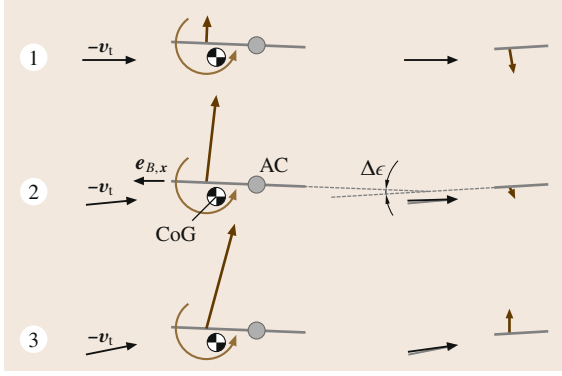


Fig. 26.20 Forces and moments at the main wing and tail for zero lift (1), the stable equilibrium AOA (2), and for high lift (3). The forces and moments are drawn into the individual surfaces' aerodynamic centers. Note the tilted inflow at the tail due to downwash. The green filled circle denotes the overall airplane aerodynamic center (AC)

Avidbots Ex1016
Page 680 of 2154

with a simplified airplane side-view as compared to the points (1) and (3), i. e., zero-lift ($\alpha = 0$) and high-lift, respectively. The stability criterion can be equivalently stated as: the airplane COG needs to be *in front* of the airplane aerodynamic center. Note the main parameters that influence the stability are tail lever arm, tail area, the *longitudinal dihedral* $\Delta\epsilon$ (Fig. 26.20 for its geometric definition), and the COG location along the x-axis.

26.4.3 Dynamic Model

While some core characteristics such as static stability and performance measures may already be established using aerodynamics coefficients only, we now turn to analyze the dynamics, since they provides a much richer insight into airplane characteristics.

For application of the 6-D (six-dimensional) rigid body dynamics (26.45), the forces and moments from the various sources need to be assembled and represented in the body frame

$$ {}^B\boldsymbol{F} = \begin{pmatrix} L\sin\alpha - D\cos\alpha + T\cos\epsilon_\mathrm{T} \\ Y \\ -L\cos\alpha - D\sin\alpha + T\sin\epsilon_\mathrm{T} \end{pmatrix} + {}^B\boldsymbol{W} \,, \tag{26.57} $$

$$ {}^B\boldsymbol{\tau} = \begin{pmatrix} L_\mathrm{A} + L_\mathrm{T} \\ M_\mathrm{A} + M_\mathrm{T} \\ N_\mathrm{A} + N_\mathrm{T} \end{pmatrix} \,, \tag{26.58} $$

with the weight in body coordinates ${}^B\boldsymbol{W} = {}^B\boldsymbol{R}_W[0, 0, -mg]^\mathrm{T}$, and where the T-subscript indicates (possible) moment components from thrust. Note that the system inputs $\boldsymbol{u}$ are hidden inside these forces and moments. Also be aware of α and β containing parts of the state vector

$$ \alpha = \arctan2(w_\mathrm{t}, u_\mathrm{t}) \,, \tag{26.59} $$

$$ \beta = \arcsin(v_\mathrm{t}/V_\mathrm{t}) \,, \tag{26.60} $$

where the true airspeed components are used

$$ {}^B\boldsymbol{v}_\mathrm{t} := \begin{cases} u_\mathrm{t} \\ v_\mathrm{t} \\ w_\mathrm{t} \end{cases} = {}^B\boldsymbol{v} - {}^B\boldsymbol{R}_W {}^W\boldsymbol{w} \,, \tag{26.61} $$

with the wind vector ${}^W\boldsymbol{w}$.

Furthermore, due to the airplane symmetry plane, the inertia matrix becomes

$$ {}^B\mathbf{I} = \begin{pmatrix} I_{xx} & 0 & I_{xz} \\ 0 & I_{yy} & 0 \\ I_{xz} & 0 & I_{zz} \end{pmatrix} \,. \tag{26.62} $$

Parametric Force and Moment Models

Let us consider the example of a simple airplane configuration with ailerons, a rudder and an elevator plus a propeller, driven by an electric motor at rotation speed ω_p.

We define the system input vector $\boldsymbol{u} = [\delta_\mathrm{a}, \delta_\mathrm{e}, \delta_\mathrm{r}, \delta_\mathrm{T}]$ as normalized aileron, rudder, and elevator action, $\delta_\mathrm{a}, \delta_\mathrm{e}, \delta_\mathrm{r} \in [-1, 1]$ as well as (normalized) thrust $\delta_\mathrm{T} \in [0, 1]$.

The fully parametric nonlinear model provided below largely follows [26.19]. We approximate the lift, drag as well as sideslip coefficients with polynomials in α and β

$$ \begin{aligned} c_\mathrm{L} &\approx c_{\mathrm{L},0} + c_{\mathrm{L},\alpha}\alpha + c_{\mathrm{L},\alpha 2}\alpha^2 + c_{\mathrm{L},\alpha 3}\alpha^3 \,, \\ c_\mathrm{D} &\approx c_{\mathrm{D},0} + c_{\mathrm{D},\alpha}\alpha + c_{\mathrm{D},\alpha 2}\alpha^2 + c_{\mathrm{D},\beta 2}\beta^2 \,, \\ c_\mathrm{Y} &\approx c_{\mathrm{Y},\beta}\beta \,. \end{aligned} \tag{26.63} $$

For many applications except slow flying airplanes, the second-order and third-order term of c_L can be omitted.

As far as the torques are concerned, we introduce also dependencies on normalized angular rates

$$ {}^B\boldsymbol{\omega}_\mathrm{n} = (p_\mathrm{n}, q_\mathrm{n}, r_\mathrm{n})^\mathrm{T} = \left(\frac{pb}{2V_\mathrm{t}}, \frac{q\bar{c}}{2V_\mathrm{t}}, \frac{rb}{2V_\mathrm{t}} \right)^\mathrm{T} \,. \tag{26.64} $$

A suitable approximation of the moment coefficients is now made as

$$ \begin{aligned} c_\mathrm{l} &\approx c_{\mathrm{l},0} + c_{\mathrm{l},\delta_a}\delta_a + c_{\mathrm{l},\beta}\beta + c_{\mathrm{l},p_\mathrm{n}}p_\mathrm{n} + c_{\mathrm{l},r_\mathrm{n}}r_\mathrm{n} \,, \\ c_\mathrm{M} &\approx c_{\mathrm{M},0} + c_{\mathrm{M},\delta_e}\delta_e + c_{\mathrm{M},\alpha}\alpha + c_{\mathrm{M},q_\mathrm{n}}q_\mathrm{n} \,, \\ c_\mathrm{N} &\approx c_{\mathrm{N},0} + c_{\mathrm{N},\delta_r}\delta_r + c_{\mathrm{N},\beta}\beta + c_{\mathrm{N},r_\mathrm{n}}r_\mathrm{n} \,. \end{aligned} \tag{26.65} $$

Finally, the propeller thrust force needs to be modeled. Using the advance ratio $J = \frac{2\pi V_\mathrm{t}}{\omega_\mathrm{p} d}$, with propeller diameter d, we can approximate the thrust coefficient

$$ c_\mathrm{T} \approx c_{\mathrm{T},0} + c_{\mathrm{T},J}J + c_{\mathrm{T},J2}J^2 \,. \tag{26.66} $$

The thrust is then obtained as

$$ T = \rho \left(\frac{\omega_\mathrm{p}}{2\pi} \right)^2 d^4 c_\mathrm{T} \,. \tag{26.67} $$

Linearized Dynamics

As common throughout literature, the linearized airplane dynamics are written using Euler angles, which is why we will follow the same approach. But conceptually, they could be written in a singularity-free form using a minimal quaternion perturbation.

Typically, a separation into *longitudinal* and *lateral* dynamics is made, in order to assess related characteristics separately. Furthermore, the state is transformed to contain α, β, and V_t rather than ${}^B\boldsymbol{v}$.

The linear dynamics around a reference state $\boldsymbol{x}_0$ and input $\boldsymbol{u}_0$ vector takes the form

$$\Delta\dot{\boldsymbol{x}}_{\text{lon}} = \boldsymbol{A}_{\text{lon}}\Delta\boldsymbol{x}_{\text{lon}} + \boldsymbol{B}_{\text{lon}}\Delta\boldsymbol{u}_{\text{lon}}$$

and

$$\Delta\dot{\boldsymbol{x}}_{\text{lat}} = \boldsymbol{A}_{\text{lat}}\Delta\boldsymbol{x}_{\text{lat}} + \boldsymbol{B}_{\text{lat}}\Delta\boldsymbol{u}_{\text{lat}} .$$

The following formulation follows [26.19] to a large extent.

Figure 26.21 describes the separation in terms of inputs $\Delta\boldsymbol{u}$ and states $\Delta\boldsymbol{x}$ for the linearized system.

The longitudinal nonlinear equations are given as

$$\begin{aligned}
\dot{q} &= \frac{1}{I_{yy}}[M_A + M_T - (I_{xx} - I_{zz})pr + I_{xz}(p^2 - r^2)] ,\\
\dot{V}_t &= \frac{1}{m}[-D\cos\beta + Y\sin\beta\\
&\quad + T\cos(\alpha - \epsilon_T)\cos\beta + mg_1] ,\\
\dot{\alpha} &= \frac{1}{\cos\beta}\left[\frac{1}{mV_t}(-L - T\sin(\alpha - \epsilon_T) + mg_3) + q_A\right] ,\\
\dot{\theta} &= q\cos\varphi - r\sin\varphi ,
\end{aligned} \tag{26.68}$$

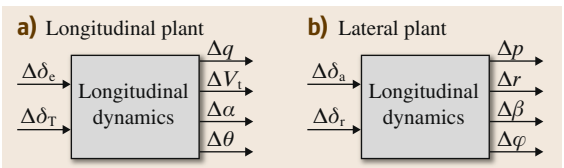


Fig.26.21a,b Linearized **(a)** longitudinal, and **(b)** lateral plants for inputs $\Delta\boldsymbol{u}$ (around $\boldsymbol{u}_0$) of the local states $\Delta\boldsymbol{x}$ (around $\boldsymbol{x}_0$)

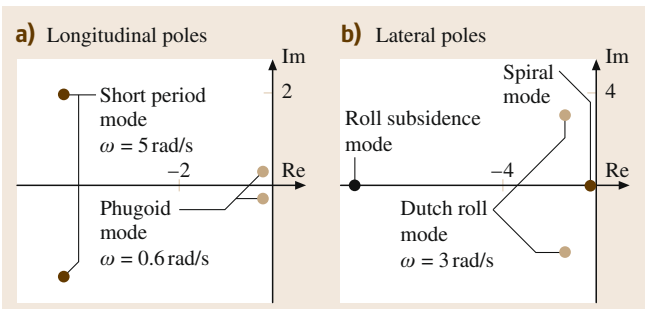


Fig. 26.22 **(a)** Longitudinal, and **(b)** lateral poles of an example aerobatic RC airplane

and the lateral nonlinear equations amount to

$$\begin{aligned}
\dot{p} &= \frac{I_{zz}(L_A + L_T - T_p)}{I_{xx}I_{zz} - I_{xz}^2} - \frac{I_{xz}(N_A + N_T - T_r)}{I_{xx}I_{zz} - I_{xz}^2} ,\\
\dot{r} &= \frac{I_{xz}(L_A + L_T - T_p)}{I_{xx}I_{zz} - I_{xz}^2} + \frac{I_{zz}(N_A + N_T - T_r)}{I_{xx}I_{zz} - I_{xz}^2} ,\\
\dot{\beta} &= -r_A + \frac{1}{mV_t}[Y\cos\beta + D\sin\beta\\
&\quad - T\cos(\alpha - \epsilon_t)\sin\beta + mg_2] ,\\
\dot{\varphi} &= p + q\sin\varphi\tan\theta + r\cos\varphi\tan\theta ,
\end{aligned} \tag{26.69}$$

where the following terms were used

$$\begin{aligned}
g_1 &= g(-\cos\alpha\cos\beta\sin\theta + \sin\beta\sin\varphi\cos\theta\\
&\quad + \sin\alpha\cos\beta\cos\varphi\cos\theta) ,\\
g_2 &= g(\cos\alpha\sin\beta\sin\theta + \cos\beta\sin\varphi\cos\theta\\
&\quad - \sin\alpha\sin\beta\cos\varphi\cos\theta) ,\\
g_3 &= g(\sin\alpha\sin\theta + \cos\alpha\cos\varphi\cos\theta) ,\\
q_A &= q\cos\beta - p\sin\beta\cos\alpha - r\sin\alpha\sin\beta ,\\
r_A &= r\cos\alpha - p\sin\alpha ,\\
T_p &= (I_{zz} - I_{yy})qr + I_{xz}pq ,\\
T_r &= (I_{yy} - I_{xx})qp - I_{xz}qr .
\end{aligned} \tag{26.70}$$

The linearizations of (26.68) and (26.69) are straightforward to obtain and not provided here due to space constraints. For a specific operating point, typically stationary ($\varphi_0 = 0$, $\theta_0 = \alpha_0$, and ${}^B\boldsymbol{\omega} = \boldsymbol{0}$), the standard tools of linear systems analysis can be employed. Most importantly, the *pole* locations in the imaginary plane will tell the characteristic modes and their dynamic stability. Figure 26.22 shows a pole location plot for an example RC airplane and introduces the related mode names.

In the case of a real pole π_i, it has the time constant $\tau_i = -1/\text{Re}(\pi_i)$. In the case of a complex conjugate pole pair, it is associated with a damping ratio $\zeta_i = -\text{Re}(\pi_i)/\sqrt{\text{Im}(\pi_i)^2 + \text{Re}(\pi_i)^2}$ and with an eigenfrequency $\omega_i = \sqrt{\text{Im}(\pi_i)^2 + \text{Re}(\pi_i)^2}$.

Figure 26.23 illustrates and characterizes the main modes.

26.4.4 Design Guidelines

Airplane design typically consists of the three stages *conceptual* design, *preliminary* design, and *detail* design. Here, we focus on the first two phases; due to space constraints, details of airplane structural design

Avidbots Ex1016
Page 682 of 2154

and analysis are not covered here. The reader is referred to respective literature, e.g., [26.20], or [26.21], the latter covering RC-type aircraft. In the following, we provide a quick overview of practical guidelines for the typically iterative design process related to achieving characteristics as described above. The guidelines follow largely [26.22] and [26.18], with focus on slow-flying small-scale UAS.

Sizing and Geometry of Main Components

In the following, we provide some rules of thumb as initial guess for the design process in terms of sizing the wing, tail, control surfaces, and the propulsion unit. As a first and very general advice, the engineer is encouraged to minimize wetted area and cross section, as well as any kind of nacelles for increased aerodynamic efficiency.

Wing. First, an existing airfoil shall be chosen with characteristics meeting the requirements in the target flow regime (Re and Ma). When it comes to determining the overall wing size, a first estimate of design weight including structure, avionics, payload, propulsion unit, and energy storage is of crucial importance (26.2). With the target speed V_r and design lift coefficient c_l, a rough guess can be made for the wing area

$$A = 2mg/(\rho V_r^2 c_l) \, .$$

Concerning wing shape, clearly highest efficiency is reached with high aspect ratios (plus no multiple lifting surfaces), and, at low Ma, no sweep-back – as long as still implementable with a structural concept and staying at reasonably high Re numbers. For high efficiency, it is advisable to achieve an elliptic lift distribution to some extent by geometry. For benign stalling characteristics, it is furthermore highly advisable to twist the wing leading edge downward with increased spanwise distance (which also influences the lift distribution).

Finally, some dihedral should be considered for roll stability.

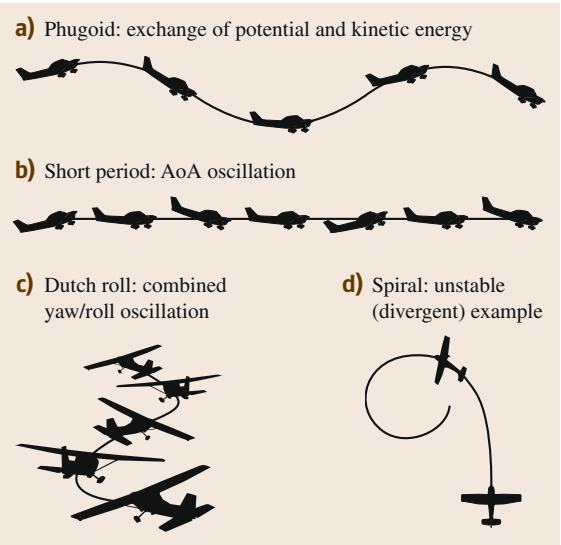


Fig.26.23a–d Characteristic trajectories of excited longitudinal modes (**a**,**b**) and lateral modes (**c**,**d**)

Tail. Various types of tails and even *exotic* configurations like the canard have been suggested; here, we want to simply point out the importance of the so-called *tail volume coefficient*, c_{VT} and c_{HT} concerning vertical and horizontal tail, respectively,

$$c_{VT} = l_{VT} A_{VT}/(bA) \, , \tag{26.71}$$

$$c_{HT} = l_{HT} A_{HT}/(\bar{c}A) \, , \tag{26.72}$$

with the wing to vertical tail lever arm l_{VT} and the wing to horizontal tail l_{HT} (these are referenced to the individual mean 1/4-chord points). Typical values for small-size slow airplanes are c_{VT}=0.02–0.04 and c_{HT} = 0.5–0.7. Furthermore, care should be taken that control surfaces are not completely blanketed in the case of stall (for stall/spin recovery).

Control Surfaces. Ailerons typically extend from around 50% in span direction to 90%; in this setting, 20–30% of wing chord is suggested as aileron depth. Tail control surface depth is typically chosen around 40% of the respective chord.

Propulsion. Finally, some advice is given concerning the propulsion unit. Some UAVs are required to be handlaunched: note that this imposes limits on the overall maximum take-off mass and minimum/stall airspeed. Experience shows that reasonable limits are <9 m/s minimum/stall speed and 7 kg airplane mass. For such small UAVs, a static thrust to weight ratio of at least 50% is highly recommended. In general, the propulsion unit must be sized to meet the specifications in terms of climb rates, maximum level flight speed and service ceiling. For the highest efficiency, the propulsion unit should be designed such as to provide highest efficiency at the design operating point (subscript r) $T_r/W = c_{D,r}/c_{L,r}$. For a hobbyist brushless DC outrunner type motor, the maximum power per motor mass ratio of 3.4 kW/kg can be used for estimation of the propulsion unit weight [26.23] (gearbox and propeller mass not included).

Handling Qualities

Manned aviation introduced the notion of *handling qualities*, assessing how well an aircraft can be flown by a human pilot as a basis for certification of both civil and military airplanes. Since UAS typically rely on autopilot systems enabling a certain degree of autonomy, these concepts may not be directly applied, but are still extremely relevant. Most importantly, the handling qualities concerning static and dynamic stability, as well as controllability determine the success of a UAS design.

While an autopilot can handle more and faster instabilities than a pilot, it can certainly not compensate for missing actuation authority. As detailed in Sect. 26.7, it is advisable to implement de facto manual operation mode as a testing, backup, or even standard operation mode, in which the airplane is either steered manually or through some stability augmentation system (SAS). Therefore, it is highly advisable that the resulting system complies with the following core requirements (simplified from [26.18]):

- Static longitudinal stability: most aft COG at least 5% of $\bar{c}$ in front of aerodynamic center (static margin, SM).
- Phugoid damping $\zeta_{ph} > 0.2$.
- Short-period oscillation $\omega_{sh} > 2$, $\zeta_s > 0.5$.
- Spiral mode may be unstable, if $\tau_{sp} > 20\,s$.
- Roll acceleration at maximum aileron deflection $|\dot{p}(\delta_{a,max})| > 5\,rad/s^2$, roll subsidence time constant $\tau_{rs} < 1\,s$.
- Dutch roll damping $\zeta_{dr} > 0.1$.
- Spins shall not be entered abruptly and must always be recoverable.

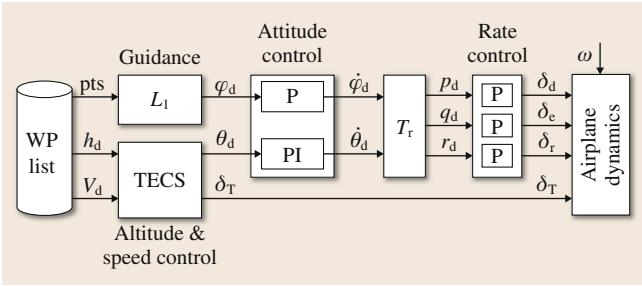


Fig. 26.24 Simple cascaded airplane guidance and control system

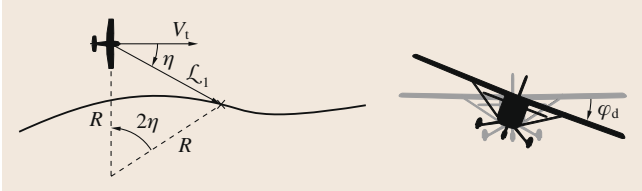


Fig. 26.25 Illustration of $\mathcal{L}_1$ lateral guidance

26.4.5 A Simple Autopilot

As shown earlier, airplane dynamics are nonlinear multiple-input-multiple-output (MIMO) systems with a significant amount of cross-coupling, thus they are inherently challenging to control. While a plethora of control strategies have been suggested as autopilots, we will provide a simple yet functional approach here that employs the popular concept of *cascaded* control loops as well as simple linear single-input–single-output (SISO) *PID* controllers acting on subparts of the dynamics. This approach is still widely deployed and well-understood, despite the fact that more advanced controllers, such as model-based linear quadratic regulators (LQR) with gain scheduling or nonlinear dynamic inversion (NDI) may achieve significantly better performance. Reference [26.24] constitutes an eccelent reference for in-depth treatment of small UAS guidance and control with cascaded control loops.

Cascaded Control Architecture

Figure 26.24 introduces the overall controller architecture. It presents the (typical) separation of per-axis *rate controllers* at the innermost loop, followed by an *attitude* controller and a combined altitude and speed controller TECS (total energy control system), as well as by a lateral $\mathcal{L}_1$ guidance. In general, care must be taken to separate successively closed loops by around a decade in terms of bandwidth.

Assuming little cross-axis sensitivity, the rate controllers may be implemented with simple individual P-controllers, optionally with $1/V_t^2$ gain scaling. Also the attitude controllers can be as simple as P and PI-controllers for roll and pitch, respectively. Note that the desired roll and pitch angle derivatives need to be transformed in the static block T_r into angular reference rates: this can be achieved by applying the inverse Jacobian $\mathbf{J}_r$ from (26.42) – where the missing yaw angle time derivative can be computed from the coordinated turn constraint $\beta = 0$, $\dot{\beta} = 0$ in (26.69)

$$\dot{\psi} = -\frac{g \sin\phi}{V_t \cos\alpha \cos\phi} + \frac{\dot{\theta}_d \sin\phi}{\cos\phi \cos\theta} \,. \tag{26.73}$$

The combined altitude and speed controller (TECS) inspired from [26.25] uses the difference to the reference altitude $\Delta h = h_d - h$ to compute a desired climb rate of the form $\dot{h}_d = \dot{h}_{traj} + K_{P,alt}\Delta h$, with the given trajectory rate of climb $\dot{h}_{traj}$ and a P-gain $K_{P,alt}$. Knowing the speed and corresponding angle of attack, $\dot{h}_d$ can be simply converted into a desired pitch angle θ_d – to be saturated according to maximum thrust (climb) and drag (sink). Since climb rate must be provided via

additional thrust, the respective power component is computed as $\Delta T_{\text{climb}} = mg\dot{h}_{\text{d}}/V_{\text{t}}$. Concerning the speed control, the second thrust component is computed with the P-gain $K_{P,\text{vel}}$ as $\Delta T_{\text{acc}} = mK_{P,\text{vel}}\Delta V_{\text{t}}$.

As a last autopilot component, the $\mathcal{L}_1$ lateral guidance [26.26] proceeds as follows (illustrated in Fig. 26.25): a reference circular path of radius R is calculated that intersects the reference path given by the waypoint sequence at look-ahead distance L_1. In order to track this reference, a centripetal acceleration of V_{t}^2/R is needed, which can now be directly translated into a desired roll angle $\varphi = \arctan(V_{\text{t}}^2/(Rg))$ corresponding to a coordinated (level) turn – saturated with maximum bank angles.

Note that the suggested scheme should be enhanced by stall prevention and recovery (AOA monitoring and control), as well as by preventing sideslip for safer (and more efficient) operation.

26.5 Rotorcraft Modeling and Design

Various types of rotorcraft UAS configurations have been developed in the past (some examples are shown in Fig. 26.26), from helicopter-type UAVs such as [26.27, 28], over a vast selection of multicopter configurations such as [26.29, 30] and tail-sitter vehicles such as [26.31, 32] up to completely new types of flight mechanisms [26.33, 34]. The design, modeling, and system analysis process for all these RW-UAS types is essentially very similar and is largely based on the methodologies originally developed within the aerospace community for full-scale rotorcraft design and evaluation [26.15, 35]. In this context, it is important to realize that the rotorcraft design process goes beyond mere efficiency and payload considerations focused on the propulsion components (e.g., using BEMT). Designing an effective RW-UAS should in principle also include flight dynamics assessments of the entire robotic flight platform.

Flight performance assessment is commonly based on one of two types of modeling approaches referred to as *quasi-steady* and *hybrid* [26.36]. The quasi-steady modeling method employs a single rigid body representation of the aircraft affected by the steady-state forces and moments originating from the propulsion subsystem. Hybrid models treat the rotorcraft as a multibody system where the dynamics of the aircraft body are coupled with additional dynamics of the rotor or propeller blades (e.g., blade flapping dynamics). For propeller-based RW-UAS like multicopter and tail-sitter vehicles using the quasi-steady approach to, e.g., model attitude dynamics is most widespread. This may be related to the fact that for these vehicle configurations, properly accounting for motor dynamics may be more relevant than accounting for high-order effects related to structural deformations of the propeller blades. For helicopter-type UAVs the hybrid approach is more common as some dynamic modes of the rotor system are likely to couple with the attitude dynamics of the main rotorcraft body. Note that a proper application of the hybrid approach is considerably more involved than using quasi-steady models and should only be resorted to if justified.

A detailed treatment of the specific modeling and design procedures for every robotic rotorcraft configuration is beyond the scope of this chapter and thus the presented considerations focus on helicopter-type and multicopter UAVs. Based on the extensive theoretical aerospace-related background available in [26.14, 15, 35] amongst others, models for the most relevant rotor respectively propeller forces and moments are presented and subsequently appended to the rotorcraft body dynamics discussed in Sect. 26.3.6. A simplified

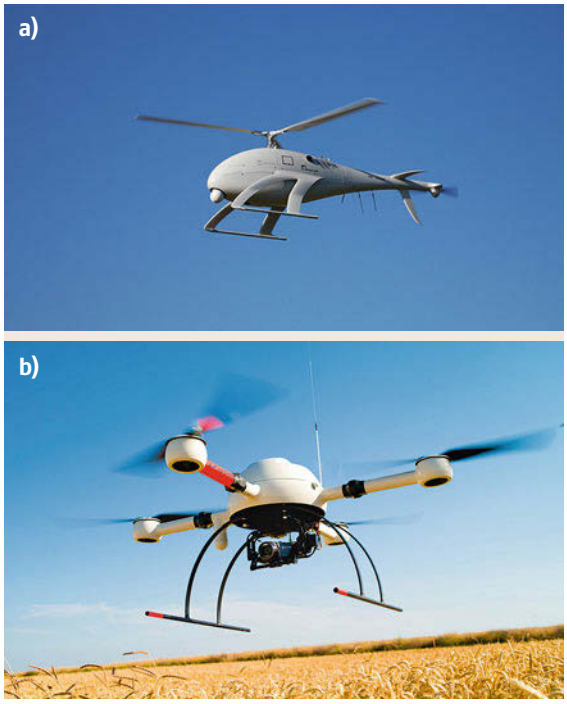


Fig. 26.26 **(a)** Conventional helicopter configuration. Swiss UAV Neo S-300. **(b)** Quadro-copter configuration. Microdrones MD4-1000

hybrid modeling approach is introduced and reductions to quasi-steady models are discussed where applicable. Finally, a few metrics meaningful for rotorcraft design and control purposes are discussed and summarized shortly.

26.5.1 Mechanical Design of Rotors and Propellers

The main control mechanism for any RW-UAS is its rotors or propellers. Accordingly, to understand the working principles and the dynamics of the rotorcraft type of aircraft, it is worth investigating a few of the main design characteristics found in these flying mechanisms.

The following discussion focuses on the operation principles of helicopter rotors and will be expanded to propellers subsequently. Figure 26.27 visualizes the typical rotor degrees of freedom realized via the *flap*, *lead-lag* and *feathering* (also referred to as *pitch*) hinges. The flap hinge allows the rotor blade to flap due to aerodynamic and inertial loads affecting the blade body during flight. The lead-lag hinge responds to lateral rotor blade moments due to Coriolis forces related to flapping. Where the flap and the lead-lag hinges are usually passive, possibly augmented with spring or damper elements, the pitch hinge is active in order to adjust the blade angle of attack and thus the generated aerodynamic forces.

Three types of rotor hubs are typically found in modern helicopters referred to as *teetering*, *articulated*, and *hingeless* depending on the mechanical realization of the flap hinge (Fig. 26.28). In the case of teetering rotor, a single hub flap hinge is located directly on the rotorshaft axis, rigidly connecting a set of two rotor blades. For the articulated rotor, the blades and the rotor hub are connected via mechanical hinges at a specific offset e from the rotor shaft axis, thus allowing each blade to flap individually. The hingeless rotor flaps through the deformation of elastic elements connecting the hub with each individual rotor blade or directly through structural deformation of the blades themselves. In this case, a virtual hinge offset can be defined at the intersection of the rotor hub plane and the tangent to the deflecting blade body at 75% of the rotor radius [26.14]. The specific characteristics of the flap hinge (offset from rotor shaft, stiffness, and damping) are fundamental for rotor blade flapping and in consequence for the rotorcraft pitch and roll dynamics as discussed later.

In the case of most propeller-based rotary-wing (RW) UAVs (e.g., multicopter systems) the feathering and lead-lag degrees of freedom do not exist. Nevertheless, propeller-based RW-UAVs may exert blade flapping by deformation of the propeller blades.

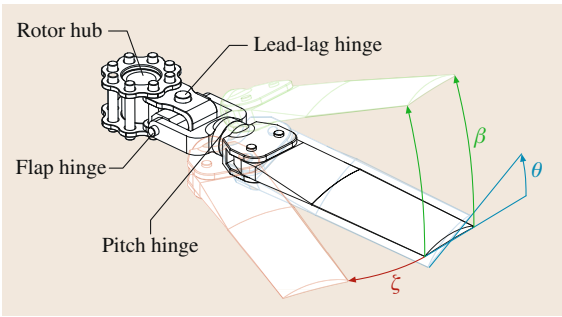


Fig. 26.27 Typical hinge configuration of an articulated rotor blade

26.5.2 Rotorcraft Dynamics

As discussed in Sect. 26.3.6, the main rotorcraft body dynamics can be directly described by the simplified differential (26.45) which for most rotorcraft systems maintain the presented mathematical structure. For helicopter-type and multicopter UAV configurations the dominant set of external forces and moments affecting these dynamics can be summarized as

$$\boldsymbol{F} = \boldsymbol{F}_{\mathrm{G}} + \sum_{i=1}^{N_{\mathrm{r}}} \boldsymbol{F}_{\mathrm{T}}^{i} + \sum_{i=1}^{N_{\mathrm{r}}} \boldsymbol{F}_{\mathrm{H}}^{i} + \boldsymbol{F}_{\mathrm{D}} \,, \tag{26.74}$$

$$\boldsymbol{\tau} = \sum_{i=1}^{N_{\mathrm{r}}} \boldsymbol{\tau}_{\mathrm{Q}}^{i} + \sum_{i=1}^{N_{\mathrm{r}}} \boldsymbol{\tau}_{\mathrm{T}}^{i} + \sum_{i=1}^{N_{\mathrm{r}}} \boldsymbol{\tau}_{\mathrm{H}}^{i} + \sum_{i=1}^{N_{\mathrm{r}}} \boldsymbol{\tau}_{\beta}^{i} \,. \tag{26.75}$$

The vector $\boldsymbol{F}_{\mathrm{G}}$ represents the weight force and $\boldsymbol{F}_{\mathrm{T}}^{i}$ is the thrust force of the i-th rotor or propeller out of the total set of N_r rotors or propellers (Fig. 26.29). These forces are related to the body heave dynamics but due

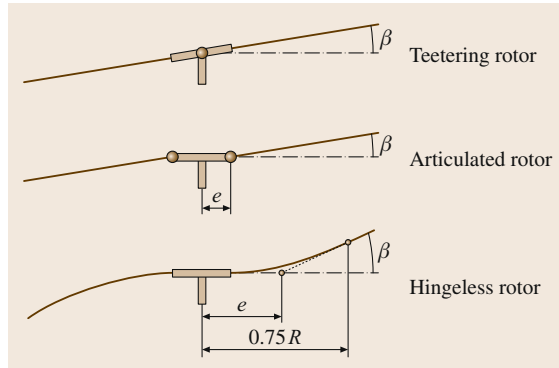


Fig. 26.28 Rotor hub design concepts of a teetering rotor, an articulated rotor with hinge offset e and a hingeless rotor with a virtual hinge offset e

to the underactuation of most rotorcraft are also responsible for the lateral rotorcraft acceleration.

The additional in-plane hub forces $\boldsymbol{F}_{\mathrm{H}}^{i}$ represent drag-related effects that may be neglected near hover but become more dominant for higher lateral flight velocities [26.15]. The vector $\boldsymbol{F}_{\mathrm{D}}$ represents the drag force associated with the rotorcraft main body. For simplicity's sake, it is assumed that the center of pressure of the rotorcraft body is collocated with its center of gravity which is not necessarily the case.

The relevant set of external moments is defined by the torques $\boldsymbol{\tau}_{\mathrm{Q}}^{i}$ affecting the vehicle yaw dynamics, the thrust-induced moments $\boldsymbol{\tau}_{\mathrm{T}}^{i}$, the moments introduced by the hub forces $\boldsymbol{\tau}_{\mathrm{H}}^{i}$ and the flapping moments related to the rotor hub stiffness $\boldsymbol{\tau}_{\beta}^{i}$.

More explicit expressions for the respective force and moment terms may be found in Tables 26.6 and 26.7, where T_i is the averaged thrust force magnitude, H_x^i and H_y^i are the hub force components along the rotorcraft body frame x- and y-axes, and Q_i is the torque generated by the i-th propeller or rotor. The aerodynamic drag of the main rotorcraft body, represented by the components D_x, D_y, and D_z, has been discussed in some detail in Sect. 26.3.5.

The vector ${}^{B}\boldsymbol{r}_i$ corresponds to the displacement of the i-th rotor or propeller hub from the rotorcraft body frame origin and ${}^{B}\boldsymbol{n}_i$ is the tip-path plane normal of the i-th propeller or rotor disk as explained in [26.37].

The coefficients β_{1c} and β_{1s} represent the longitudinal and lateral flapping coefficients [26.15] describing the tilting of the rotor or propeller disk as elaborated in more detail later in this section. Finally, the parameter k_b^i corresponds to the flapping spring stiffness of the i-th rotor or propeller hub. In the case of articulated or teetering hubs this torsional spring stiffness represents potential flap hinge springs and in case of hingeless hubs approximates the structural bending stiffness of a particular rotor or propeller.

In the case of a helicopter tail rotor, flapping is usually neglected and the thrust direction is modeled as, e.g.,

$$ {}^{B}\boldsymbol{n}_i = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} . \tag{26.76} $$

To finalize the models of these external forces and moments, the respective aerodynamic effects as well as the role and characteristics of blade flapping must be discussed next.

Table 26.6 Typical rotorcraft forces

Type of force	Expression
Gravity	${}^{B}\boldsymbol{F}_{\mathrm{G}} = {}^{B}\boldsymbol{R}_{W} \begin{pmatrix} 0 \\ 0 \\ -mg \end{pmatrix}$
Thrust	${}^{B}\boldsymbol{F}_{\mathrm{T}}^{i} = {}^{B}\boldsymbol{n}_i T_i$
Hub forces	${}^{B}\boldsymbol{F}_{\mathrm{H}}^{i} = \begin{pmatrix} H_x^i \\ H_y^i \\ 0 \end{pmatrix}$
Rotorcraft body drag	${}^{B}\boldsymbol{F}_{\mathrm{D}} = \begin{pmatrix} D_x \\ D_y \\ D_z \end{pmatrix}$

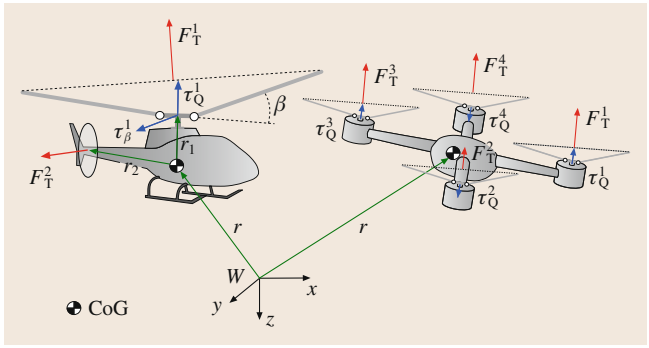


Fig. 26.29 A conventional helicopter and a quadro-copter platform. Flapping angles have been exaggerated and only the rotor and propeller forces relevant near hover have been visualized

26.5.3 Simplified Aerodynamics

In order to accurately predict the forces and moments generated by a rotorcraft system, accounting for various ranges of operation conditions, detailed design specifications, aerodynamic interactions between the different rotors or propellers and possibly the rotorcraft body itself, highly sophisticated aerodynamics simula-

Table 26.7 Typical rotorcraft moments

Type of moment	Expression
Torque	${}^{B}\boldsymbol{\tau}_{\mathrm{Q}}^{i} = \begin{pmatrix} 0 \\ 0 \\ Q_i \end{pmatrix}$
Thrust moment	${}^{B}\boldsymbol{\tau}_{\mathrm{T}} = {}^{B}\boldsymbol{r}^{i} \times {}^{B}\boldsymbol{f}_{\mathrm{T}}^{i}$
Hub moment	${}^{B}\boldsymbol{\tau}_{\mathrm{H}}^{i} = {}^{B}\boldsymbol{r}^{i} \times {}^{B}\boldsymbol{f}_{\mathrm{H}}^{i}$
Flap moment	${}^{B}\boldsymbol{\tau}_{\beta}^{i} = k_b^i \begin{pmatrix} -\beta_{1s}^i \\ \beta_{1c}^i \\ 0 \end{pmatrix}$

Avidbots Ex1016
Page 687 of 2154

tion tools are required. Such tools are usually not easily accessible to or operated by the roboticist and may hide some of the fundamental flight dynamics characteristics of a particular platform due to the involved complexity. For a repeated evaluation of the general flight properties of a rotorcraft it may thus be preferable to derive approximate, analytical models of the aerodynamic forces and moments. This may provide profound insight into the core working principles of rotorcraft operation.

The general approach to derive such models is the employment of BET as partially presented in Sect. 26.3.4. BET, as opposed to BEMT or MT, maintains the notion of individual rotor or propeller blades. Hence, it may account for changes in the angle of attack α, the aerodynamic inflow velocities U_T and U_P and in consequence of the local lift and drag increments $\mathrm{d}L$ and $\mathrm{d}D$, not only in function of the radial position r of the observed airfoil segment but also in dependency of the blade azimuth ξ. To further the understanding of this dependency in ξ, one may examine Fig. 26.16 as well as Fig. 26.30 where the hub coordinate frame H is introduced.

The perpendicular and tangential inflow velocities U_P and U_T (and thus also α) are strongly related to the rotor angular speed Ω and the induced inflow velocity v_i (as discussed for BEMT). In flight operation beyond hover, these inflow velocities are also affected by the rotorcraft linear and angular velocities ${}^B\boldsymbol{v}$ and ${}^B\boldsymbol{\omega}$, as well as the potential flapping motion of the rotor or propeller blades. From the perspective of a revolving rotor or propeller blade, these additional inflow velocities vary periodically in function of ξ.

Inspecting Fig. 26.30, one can establish the following relations for the airfoil inflow velocities [26.35]

$$U_\mathrm{T} = \Omega(e+r) - v_y + \beta\omega_x r\,, \tag{26.77}$$

$$U_\mathrm{P} = v_i - w + \omega_y(e+r) + \dot{\beta}r - \beta v_x\,. \tag{26.78}$$

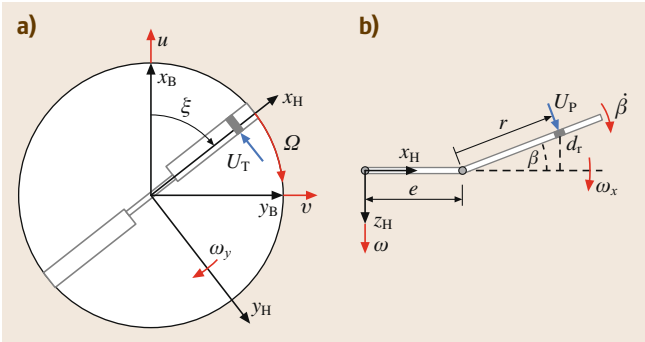


Fig. 26.30 **(a)** Top view of revolving rotor/propeller blade and the relevant velocities. **(b)** Side view of revolving rotor/propeller blade and the relevant velocities

The linear and angular velocities v_x, v_y, ω_x, and ω_y correspond to the projections of the vehicle body velocities u, v, p, and q into the hub frame H

$$v_x = u\cos\xi + v\sin\xi\,, \tag{26.79}$$

$$v_y = v\cos\xi - u\sin\xi\,, \tag{26.80}$$

$$\omega_x = p\cos\xi + q\sin\xi\,, \tag{26.81}$$

$$\omega_y = q\cos\xi - p\sin\xi\,. \tag{26.82}$$

For helicopter-type UAVs the azimuth dependency of α may also be related to periodic changes in the blade pitch angle θ. For helicopter configurations the feathering angle θ can usually be controlled collectively for the entire rotor disk or cyclically in dependency of the blade position ξ using a swashplate mechanism [26.38]

$$\theta = \theta_0 + \theta_{1\mathrm{c}}\cos\xi + \theta_{1\mathrm{s}}\sin\xi\,. \tag{26.83}$$

The swashplate mechanism essentially provides the means to adjust the collective pitch angle θ_0 as well as the cyclic pitch angles $\theta_{1\mathrm{c}}$ and $\theta_{1\mathrm{s}}$ individually. Note that for twisted propeller or rotor blades θ_0 merely represents the pitch angle at the blade root and an additional term accounting for the radial variation in pitch must be introduced [26.14].

By altering θ_0, the swashplate provides some control over the average angle of attack of the entire rotor disk and thus the generated average thrust and torque. Similarly, varying the cyclic components creates a lift imbalance between opposing sides of the rotor disc. This lift imbalance induces a periodic flapping motion of the individual rotor blades (also depending on ξ) which affects the pitch and roll moments $\boldsymbol{\tau}_\beta$ and $\boldsymbol{\tau}_\mathrm{T}$. For helicopter-type UAVs this is the main control mechanism for the pitch and roll attitude dynamics.

For the majority of multicopter vehicles θ_0 is usually fixed and the cyclic angles do not exist ($\theta_{1\mathrm{c}} = \theta_{1\mathrm{s}} = 0$). In this case, attitude control is realized through $\boldsymbol{\tau}_\mathrm{T}$ only by changing the propeller speed Ω_i of the individual propellers.

To describe the aforementioned periodic changes in blade flapping the following representation of the flapping angle β is customary when modeling rotorcraft dynamics [26.39]

$$\beta = \beta_0(t) + \beta_{1\mathrm{c}}(t)\cos\xi(t) + \beta_{1\mathrm{s}}(t)\sin\xi(t)\,. \tag{26.84}$$

The coefficient $\beta_0(t)$ corresponds to the coning angle of the entire rotor or propeller disc, whereas $\beta_{1\mathrm{c}}(t)$ and $\beta_{1\mathrm{s}}(t)$ are the longitudinal and lateral disk tilting angles.

Avidbots Ex1016
Page 688 of 2154

Unlike the blade pitch coefficients in (26.83), the flap coefficients in (26.84) are assumed to be functions of time t which is later exploited when modeling flapping dynamics.

Although it is crucial to capture all these periodic dependencies in ξ, it can be obstructive for the purpose of creating an efficient rotorcraft simulation. Directly simulating the body dynamics (26.45) under the influence of the external forces and moments (26.74) and (26.75) while accounting for all dependencies in ξ, would require the blade motion to be simulated step by step while moving around the rotor or propeller shaft. This can lead to comparatively stiff differential equations which may be hard to simulate efficiently.

An alternative approach is to derive simplified force and torque models by integrating the lift and drag increments along the rotor or propeller radius, summing over the number of blades N_b and averaging around the rotor azimuth as

$$T = \frac{N_\mathrm{b}}{2\pi}\int_0^{2\pi}\int_0^R \mathrm{d}L\,, \tag{26.85}$$

$$Q = \frac{N_\mathrm{b}}{2\pi}\int_0^{2\pi}\int_0^R r\left(\mathrm{d}D + \frac{U_\mathrm{P}}{U_\mathrm{T}}\mathrm{d}L\right)\,, \tag{26.86}$$

$$H_x = \frac{N_\mathrm{b}}{2\pi}\int_0^{2\pi}\int_0^R \sin\xi\left(\mathrm{d}D + \frac{U_\mathrm{P}}{U_\mathrm{T}}\mathrm{d}L\right)\,, \tag{26.87}$$

$$H_y = \frac{N_\mathrm{b}}{2\pi}\int_0^{2\pi}\int_0^R \cos\xi\left(\mathrm{d}D + \frac{U_\mathrm{P}}{U_\mathrm{T}}\mathrm{d}L\right)\,. \tag{26.88}$$

Combining (26.77) to (26.88) with (26.12), (26.13), (26.27), and (26.33) and assuming simple rotor or propeller blade geometries (e.g., linear blade twist) the above expressions can be evaluated algebraically, e.g., using a symbolic computation program. As opposed to BEMT these averaged aerodynamic formulations are capable of accounting for the effects of the periodic changes of inflow velocities such as, e.g., the *dissymmetry of lift* phenomenon [26.15] as well as for the varying blade feathering and flapping angles.

To gain some appreciation of how these averaged models may be useful in providing insight into the core characteristics of rotorcraft operation, a strongly simplified thrust model directly resulting from (26.85) under the assumption of hover and negligible flapping angles is given as an example

$$T = \left(k_{\mathrm{T}1}\theta_0 + k_{\mathrm{T}2}\frac{v_0}{\Omega} + k_{\mathrm{T}3}\right)\Omega^2\,. \tag{26.89}$$

The coefficients k_{T1}, k_{T2}, and k_{T3} essentially depend on aerodynamic and geometric rotor or propeller properties. As one may deduce directly from (26.89), thrust is a linear function of the collective pitch angle θ_0 and the average induced rotor or propeller inflow velocity v_0 which is further defined in the next section.

26.5.4 Nonuniform Inflow

In order to account for the varying induced flow field over the rotor or propeller disk in hover and forward flight, the following approximate induced inflow distribution may be assumed as discussed in [26.14]

$$v_\mathrm{i} = v_0 + \frac{r}{R}(v_{1\mathrm{c}}\cos\xi + v_{1\mathrm{s}}\sin\xi)\,. \tag{26.90}$$

The velocity component v_0 represents the average induced inflow at the center of the modeled rotor respectively propeller disc. It is directly related to the thrust level the rotor or propeller is operating at and may be computed, e.g., based on the iterative method discussed in [26.15]. The inflow coefficients v_{1c} and v_{1s} account for changes in the induced flow field due to lateral rotorcraft flight velocities. Various steady-state models for $v_{1\mathrm{c}}$ and $v_{1\mathrm{s}}$ have been proposed in the past with moderate prediction quality only.

Reference [26.40] presents these models in detail and discusses an alternative method which also accounts for transient effects. Accounting for inflow dynamics, the transient response of v_0, $v_{1\mathrm{c}}$, and $v_{1\mathrm{s}}$ to, e.g., sudden blade pitch changes can be captured and the aforementioned iterative computations may be avoided.

26.5.5 Flapping Dynamics

The dynamics of rotor or propeller flapping corresponds to a second-order differential equation of the flapping angle β which essentially represents an aerodynamically damped oscillator [26.35]

$$\ddot{\beta} = \ddot{\beta}(\xi, \Omega, u, v, w, p, q, \dot{p}, \dot{q}, \beta, \dot{\beta}, \theta)\,. \tag{26.91}$$

Reference [26.41] and references therein elaborate the modeling process required to derive the above differential equation from first principles.

As for the simplified aerodynamic forces and moments, the dependency of (26.91) on rotor azimuth is important but problematic for efficient blade flap simulations. Here, the averaging operation discussed in [26.39] has been employed. This essentially leads to the following differential equation of the flapping coef-

ficients β_0, β_{1c} and β_{1s}

$$A_{\ddot{\beta}}\ddot{\boldsymbol{\beta}} + A_{\dot{\beta}}\dot{\boldsymbol{\beta}} + A_{\beta}\boldsymbol{\beta} = A_{\theta}\boldsymbol{\theta} + A_{\omega}\boldsymbol{\omega} + A_{\upsilon}\boldsymbol{\upsilon} \,,$$

$$\boldsymbol{\beta} = \begin{pmatrix} \beta_0 \\ \beta_{1c} \\ \beta_{1s} \end{pmatrix} , \qquad \boldsymbol{\upsilon} = \begin{pmatrix} \upsilon_0 - w \\ \upsilon_{1c} \\ \upsilon_{1s} \end{pmatrix} ,$$

$$\boldsymbol{\omega} = \begin{pmatrix} p \\ q \\ \dot{p} \\ \dot{q} \end{pmatrix} , \qquad \boldsymbol{\theta} = \begin{pmatrix} \theta_0 \\ \theta_{1c} \\ \theta_{1s} \end{pmatrix} . \tag{26.92}$$

The involved matrices from $A_{\ddot{\beta}}$ to A_{υ} depend on an abundance of aerodynamic, geometric, inertial, and structural parameters of the modeled rotor or propeller system as well as on the lateral flight velocity of the rotorcraft. The structure of these matrices is thus strongly related to the type of rotor or propeller hub that is being modeled. From the resulting mathematical relations it is for example possible to conclude that for a teetering rotor without hinge springs, the maximal flapping response β to a cyclic pitch input in θ will follow with a 90° phase shift in rotor azimuth [26.15].

Considering the body time constants of most rotorcraft configurations in comparison with the time constants of the above dynamics, one may in general assume that the fast poles in (26.92) are negligible and they mostly represent rotor or propeller vibrations. Accordingly, (26.92) is often reduced to a first-order differential equation by introducing the assumption $\ddot{\boldsymbol{\beta}} = \mathbf{0}$. For very rigid rotors or propellers one may even assume $\dot{\boldsymbol{\beta}} = \mathbf{0}$ and compute the steady-state flapping response only (quasi-steady model). However, this is not the case for all RW-UAS configurations. To what degree higher order effects such as flapping or inflow dynamics have to be accounted for, depends on the level of frequency separation between the body poles versus the poles of the rotor dynamics.

26.5.6 Flight Dynamics Assessment

Assembling the dynamics of the rotorcraft body with the forces and moments defined in the previous section, as well as possibly introducing higher order effects such as blade flapping and inflow dynamics, results in a set of nonlinear differential equations representing the flight characteristics of the modeled platform. In order to inspect these flight characteristics one may implement a flight simulator based on this set of nonlinear differential equations and analyze the simulated flight responses. The linearization process presented in Sect. 26.3.6 may provide additional conclusions and is often followed by a separation into subsystems analyzed individually.

One of the most crucial rotorcraft subsystems corresponds to the pitch and roll attitude dynamics as it is this subsystem which ultimately defines how agile, stable, or accurate a rotorcraft may be able to fly in the horizontal plane of motion. Assuming near hover operation and neglecting higher order dynamics, the linearized open-loop roll dynamics for example, can usually be reduced to a first-order transfer function

$$G(s) = \frac{P(s)}{U(s)} = \frac{K_{\mathrm{p}}}{s - D_{\mathrm{p}}} \,. \tag{26.93}$$

The frequency function $P(s)$ represents the Laplace transform of the rotorcraft roll rate p and $U(s)$ the Laplace transform of a control input affecting the roll subsystem. For a helicopter, this control input commonly corresponds to a cyclic swashplate command and for a quadro-copter to the differential speed between two propellers on opposing sides of its airframe. The abstracted system parameter K_{p} corresponds to the so-called *roll control derivative* and D_{p} to the so-called *roll damping* [26.37]. The control sensitivity K_{p} defines how strong the initial rotorcraft roll acceleration $\dot{p}$ will respond to a control input represented by $U(s)$. The parameter D_{p} defines how well the system's dynamic response will be damped in the following. For hybrid rotorcraft models the corresponding transfer functions are generally of higher order but are equally useful when analyzed with the well-known tools of linear system theory [26.15, 35].

In general, these transfer functions and the resulting metrics such as, e.g., K_{p} and D_{p} can be derived in dependency of a specific subset of physical rotorcraft parameters. Using these simple metrics and including the knowledge that may be gathered from the nonlinear flight simulator, flight performance trends can be assessed depending on the parameters of the UAV configuration under investigation. Such parameters may for example include the body pitch and roll inertia, the location of the body center of gravity or the location of the rotors or propellers. The fundamental understanding gained in this evaluation process is crucial for the development of effective robotic flight systems and the required control laws.

Avidbots Ex1016
Page 690 of 2154

26.6 Flapping Wing Modeling and Design

A variety of animals, from insects to birds, are capable of flight maneuvers which are presently impossible in micro aerial vehicles, such as flying in turbulence or cluttered airspace. Additionally, animals are more maneuverable and can fly longer distances. People have made many attempts at building flapping robots or ornithopters. While several are successful, many either never take off or fly only for a short duration due to their higher complexity or poor design. Until recently, ornithopters represented a niche of flying vehicles. The development of lithium polymer batteries produced a light-weight high-power energy resource to power ornithopters. Among the first successful electric ornithopters were the Caltech and Aerovironment microbats in 1998 [26.42, 43]. Many designs still fail to fly despite the rapidly increasing population building electric ornithopters. A major problem in most designs is an inability to generate enough lift to take off in the first place. This precludes additional flight research, such as maneuverability, flight distance or time. Engineers have believed that flapping wings are essential to further development of micro aerial vehicles since the first electric ornithopters took off and biologists started to understand the aerodynamics of flapping insect wings. The main reason behind this focus is the idea that they are aerodynamically more efficient at the small Reynolds number of insects (10−10 000) when viscosity effects start to dominate airflow.

26.6.1 Aerodynamic Mechanisms

Our understanding of insect aerodynamics provides us with the most detailed model of the aerodynamic function of a flapping wing [26.45]. There is some evidence that wing flexibility can improve aerodynamic performance of a flapping wing by roughly 10% [26.46] if the angle of attack is not optimized for a stiff wing. However, a parametric study using a robot model of an insect wing suggests that wing flexibility does not improve performance if we can optimize angle of attack independently of wing stiffness [26.47]. Ignoring aeroelastic effects that change angle of attack distribution, the key known aerodynamic mechanisms of a flapping wing are [26.45]:

1. A stable leading edge vortex (LEV) that enables the wing to operate at high angles of attack without stall during the quasi-steady mid-stroke phase (Fig. 26.31). During stroke reversal the aerodynamics is not quasi steady. In this phase, five additional affects are thought to be important:
2. *Added mass* effects due to fluid acceleration in response to the reversal.
3. The Wagner effect explaining that changes in vortex strength need time to build-up over a few chord lengths of travel.
4. Rotational lift due to the timing of changes in angle of attack during stroke reversal and its effect on vortex lift through the *Kramer effect*.
5. Wake capture when the wing reverses direction and interacts with the momentum jet of its shed wake.
6. Clap and fling when the wings become close enough to (nearly) touch and air is forced out of the cavity formed by the two wings and sucked back in, which can increase lift [26.48].

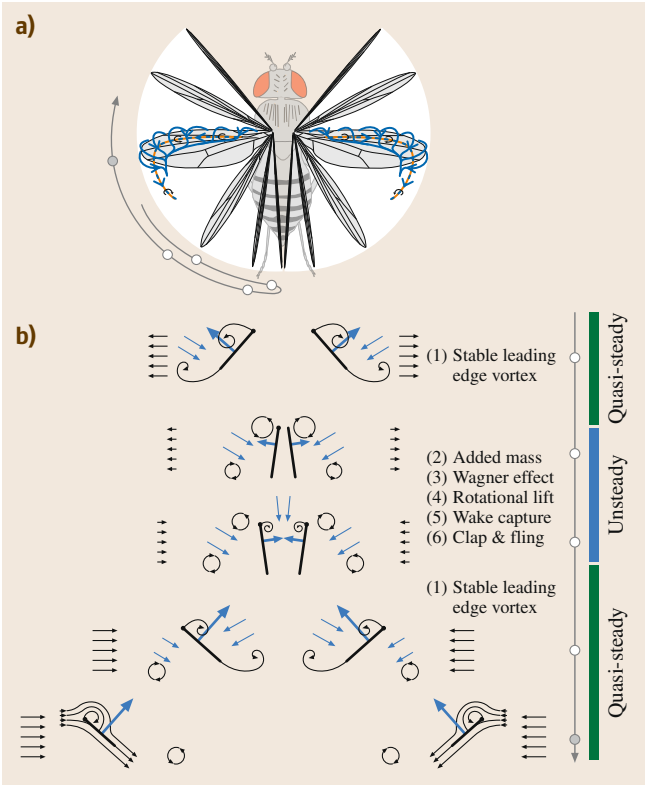


Fig.26.31a,b Flapping insect wing aerodynamics can be understood through the interaction of a myriad of complex aerodynamic mechanisms. **(a)** The key high-lift mechanism insects employ, is a stable leading edge vortex (LEV) generated during the up and downstroke. **(b)** A flapping cycle consists of a quasi-steady part during which the wing accelerates little. During this phase, the stable LEV is the key high-lift mechanism (1). During stroke reversal there is evidence that up to five effects (2)–(6) could be important (after [26.44])

Avidbots Ex1016
Page 691 of 2154

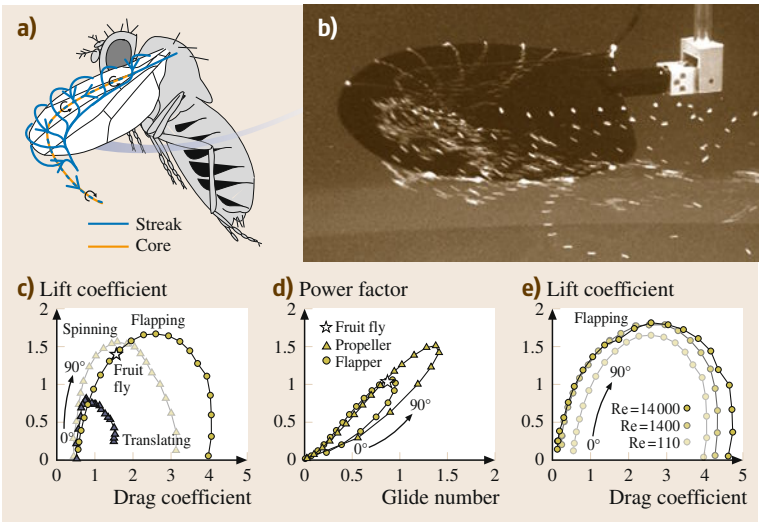


Fig.26.32a–e The aerodynamics of a flapping (insect) wing scale from insect to bird scale. **(a)** A stable LEV enables flapping wings to operate at high angles of attack without stall. **(b)** The key parameter explaining LEV stability is the wing's swing, its spinning motion, as demonstrated by this spinning model of a fly wing which generates a stable LEV and similarly elevated forces as in flapping wings. **(c)** At insect scale fixed (translating) wings underperform, whereas flapping and spinning wings generate similarly high lift. Spinning wings generate less drag which makes them more efficient. **(d)** The power factor of a spinning wing is higher than for a flapping wing, higher indicating that less power is needed to support body weight. **(e)** The dimensionless lift and drag averaged over a full flapping cycle is independent of scale to within good approximation (Reynolds number 110: fruit fly: 1400; house fly: 14 000; hummingbird). This makes flapping wing aerodynamics scalable enabling the use of dimensional analysis [26.49]

There exist, however no quantitative experimental studies or theories that fully dissect these effects and quantify their relative importance for aerodynamic lift and power. While flapping wing aerodynamics is complex and not fully understood, it is simple from a robot design perspective, because it is scalable from insect to bird size (Fig. 26.32). This enables prototyping at larger, more cost effective, scales and enables scaling the design down as technology advances, and smaller components and fabrication methods become available [26.49]. Flapping wings generate more lift than translating wings because they generate a stable LEV. To generate a stable vortex over the whole wing, the aspect ratio with respect to the center of rotation needs to be equal to or smaller than about 4 [26.53]. Flapping wings with an aspect ratio larger than 4 can stall outboard [26.53]; whereas more stubby flapping wings cannot. This can explain why the majority of insect, bird, and bat wings have an aspect ratio of around 2–4 with respect to the *shoulder* joint [26.53]. The



	Delfly II	Nano-Hummingbird	RoboBee
Wingspan (cm)	28	16	3
Mass (g)	16	19	0.06
m/m_0	1.26	1.37	N/A (tethered)
Flight time	15 min.	11 min.	N/A (tethered)
Frequency (Hz)	14	30	110
Mechanism	Gearbox and 4-bar	Gearbox and string rollers	Piezo-electric Elastic 4-bar like
Scale (mm)	10^2–10^0	10^2–10^0	10^2–10^{-1}
Power	1.4 W	3.27 W	N/A (tethered)
Current	380 mA	880 mA	N/A (tethered)

Fig.26.33a–c Examples of three different types of successful flappers. **(a)** Jaap Oldenkamp, Delfly II (after [26.49, 50]), **(b)** Nano Hummingbird (after [26.51]), **(c)** RoboBee (after [26.52])

Avidbots Ex1016
Page 692 of 2154

main advantage of stubby wings is that they do not stall at high angles of attack enabling animals to take-off and land vertically by increasing angle of attack instead of flapping frequency [26.53] using LEVs [26.54]. Insects [26.55], bats, hummingbirds [26.56], and other birds [26.57], but also auto rotating seeds generate stable LEVs. This shows that stable LEVs are a convergent evolutionary solution for high lift at high angle of attack in nature [26.53].

Comparison of flapping versus spinning (propeller-like) insect wings shows spinning insect wings generate similar elevated lift forces by generating a LEV at lower drag. Helicopters with stubby rotors are, therefore, aerodynamically more efficient than stubby flapping wings, because they need less power to fly, as qualitatively presented in Fig. 26.32d [26.49]. This is confirmed experimentally for the most advanced hovering ornithopter at present, the Nano Hummingbird [26.51]. Comparing its flapping wing with a spinning wing showed for various forward speeds that flapping wings require more power for the same lift, in part due to aerodynamics [26.49, 53], and in part due to inertia losses [26.49, 51]. The key advantage of flapping wings seems to be the potential for extreme maneuverability and robustness. For instance, flapping wings may fare better in turbulence, close to the ground, near vertical surfaces and through clutter, when helicopters can become unstable due to stall and complex rotor-wake interactions [26.58].

26.6.2 Sizing New Flappers

An improved understanding of the detailed aerodynamics is scientifically invaluable, but perhaps not critical for designing successful ornithopters at a time when most struggle to take-off. Instead, sizing an ornithopter in terms of gross design parameters such as wing span, weight, and flapping frequency is more critical for take-off. The design methodology introduced here explains how one can transform successful designs to meet other mission perspectives. These designs can then enable flight studies that can advance our understanding of ornithopters versus Ro-UAS and FW-UAS to better appreciate their unique advantages.

Amongst successful flappers, there are three main archetypes as shown in Fig. 26.33. Historically, most flappers have relied on variants of a four-bar mechanism to generate the flapping motion which generates lift. One example of this is the Delfly family of ornithopters, which are capable of both fast forward flying and hover using this approach (VIDEO 493). A recent design which demonstrates both prolonged hovering flight and maneuverability, although lacks the ability to fly fast forward, is the Aerovironment Nano-Hummingbird [26.51]. The Nano-Hummingbird uses a flapping mechanism composed of rollers and strings, while still using a geared down motor to provide power at the right frequency. Additionally, the wings provide control, rather than traditional tail control surfaces. Another more modern development is centimeter scale ornithopters which use piezoelectric actuators to generate flapping motion and control such as the Harvard Robobee [26.52] and the Berkeley Micromechanical Flying Insect. These are capable of tethered flight only, because no batteries exist that can supply high enough power in an enough lightweight package.

Despite the differences in design, these flappers share common trends in parameters, as shown in Fig. 26.34. To design a functional ornithopter, we start with a desired mission such as surveillance, search and rescue, or military applications. The mission determines an appropriate wingspan, and also determines a minimum time for task completion. Figure 26.34 shows that empty weight (mass without battery) follows an exponential pattern with wingspan, especially over the mid-range of wingspans. The main observation is that the power defining scale is not 3, but approximately 1.5. This may be because significant portions of the mass of smaller ornithopters comes from electronics, gearboxes and actuators, whose masses are not dependent on wingspan. Additionally, required flapping frequency decreases with wingspan, enabling an approximation of required flapping frequency based on wingspan that works well for all sizes of ornithopters, as expected using scaling relations.

Using initial design parameters from a successful ornithopter, we can design another ornithopter that is also capable of flapping flight using scaling relationships of geometry, fluid mechanics, and battery physics [26.27]. We need to decide on design parameters for the new flapper, including the wingspan b, weight W, aspect ratio Λ, and battery weight W_{batt}. Here, the aspect ratio is wingspan divided by chord length, as these are both easily measured design parameters. Example of initial parameters for the Delfly II are: $b_1 = 28\,\text{cm}$, $m_1 = 16\,\text{g}$ ($W = \text{mg}$), $\Lambda_1 = 3.5$, $f_1 = 14\,\text{Hz}$, $P_1 = 1.4\,\text{W}$, $W_{\text{batt},1} = 2.7\,\text{g}$, $t_1 = 15\,\text{min}$. Initial design parameters are denoted with subscript 1; while new design parameters are denoted with subscript 2. Using the curve fitted through successful ornithopters as shown in Fig. 26.34, one can make an initial approximation of empty weight. First, we can calculate the wing area, A_{fl}, of the new flapper and the old flapper using the same equation for each

$$A \propto \frac{b^2}{\Lambda} \,. \tag{26.94}$$

Avidbots Ex1016
Page 693 of 2154

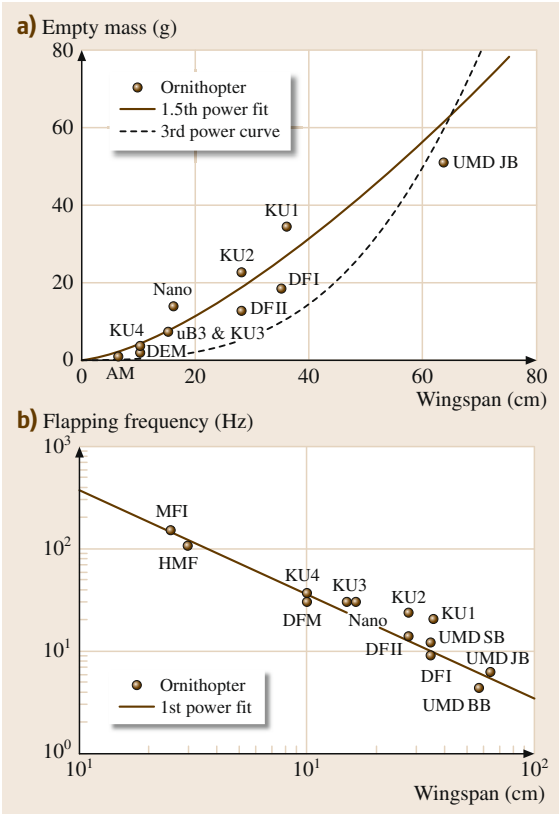


Fig.26.34a,b Current ornithopter trends of empty mass and flapping frequency with changes in wingspan. **(a)** The empty mass of successful ornithopters does not scale with wingspan cubed, but with wingspan to the power 1.5 ($R^2 = 0.79$). The power law predicts the approximate masses effectively in the 10–50 cm wingspan range, while it overestimates the mass for those with wingspans below 10 cm. The curve to the third power consistently underestimates the unloaded masses of current ornithopters. **(b)** To support the weight of the ornithopter, flapping frequency needs to increase inverse to wingspan for smaller wingspans. Ornithopters in **(a)** fly freely and have a flight time of at least one minute. The Micromechanical Flying Insect and Harvard Robobee follow the same trend line for flapping frequency as larger ornithopters; even though they fly tethered (they would need to flap faster with batteries onboard). The relationship here fits a power curve with the exponent equal to −1.01 with $R^2 = 0.96$. Abbreviations are as follows: MFI – Berkeley Micromechanical Flying Insect; HMF – Harvard Robobee; KU1,2,3,4 – Konkuk University ornithopters; DFI,II,M-Delfly I,II and Micro; Nano – Aerovironment Nano-Hummingbird; UMD SB, JB, BB – University of Maryland Small Bird, Big Bird, Jumbo Bird; AM – Brian's Ornithopter; uB3 – NiCad powered Caltech Microbat

In hovering or steady forward flight, it is reasonable to assume that weight is proportional to lift

$$W \propto \frac{1}{2} c_\mathrm{L} \rho V_\mathrm{t}^2 A_\mathrm{fl} \,. \tag{26.95}$$

We assume that c_L (lift coefficient), ρ (density,) and g (gravitational acceleration) are constant [26.49], which is reasonable for flights on earth at low altitudes. Then, rearranging produces the following relationship between forward velocities, V_t

$$V_{\mathrm{t},2} = V_{\mathrm{t},1} \sqrt{\frac{W_2}{W_1} \frac{A_{\mathrm{fl},1}}{A_{\mathrm{fl},2}}} \,. \tag{26.96}$$

We can then assume that the advance ratio J is constant for both vehicles, which is a reasonable approximation for ornithopters with similar wing kinematics, shape, and deformation. The advance ratio J is the ratio of maximum forward speed to wingtip speed

$$J = \frac{V_\mathrm{t}}{4 f \Phi R} \,. \tag{26.97}$$

Since wingspan is twice the radius, and we can use the assumption that J is constant to obtain the following relationship for flapping frequencies

$$f_2 = \frac{V_{\mathrm{t},2}}{V_{\mathrm{t},1}} \frac{b_1}{b_2} \frac{\Phi_1}{\Phi_2} f_1 \,. \tag{26.98}$$

Then, assuming that flapping amplitude, Φ is constant between the two designs (reasonable for designs that follow the same parameters and keep the same gearboxes) we can simplify the relationship for flapping frequencies

$$f_2 = \frac{V_{\mathrm{t},2}}{V_{\mathrm{t},1}} \frac{b_1}{b_2} f_1 \,. \tag{26.99}$$

The required power to fly is proportional to the weight and flight speed

$$P \propto m g V_\mathrm{t} = W V_\mathrm{t} \,. \tag{26.100}$$

Thus, we can calculate the power required of the new flapper relative to that of the old flapper

$$P_2 = P_1 \frac{V_{\mathrm{t},2}}{V_{\mathrm{t},1}} \frac{W_2}{W_1} \,. \tag{26.101}$$

Using the power calculated above, the flight time can be estimated as

$$t = \frac{C_\mathrm{LiPo} U_\mathrm{LiPo}}{P} m \,, \tag{26.102}$$

Avidbots Ex1016
Page 694 of 2154

in which $U_{\mathrm{LiPo}} = 3.7\,\mathrm{V}$ for a LiPo battery, and where, as in Fig. 26.35, the capacity can be approximated as

$$C_{\mathrm{LiPo}} = m_{\mathrm{batt}} k_{\mathrm{batt}} \,. \tag{26.103}$$

From the scaling equations (particularly (26.101) and (26.102)), we can produce a set of graphs as in Fig. 26.36, allowing us to use the wingspan and flight time to design a scaled ornithopter. Beginning with the approximate wingspan and flight time desired, we use Fig. 26.36a to choose the appropriate battery mass. An increase in wingspan creates the option for heavier batteries and an increase in flight time as does an increase in battery mass. The wingspan and battery mass specify the required flapping frequency. This allows us to choose a motor and gear ratio. If this turns out to be impractical with available components, we can adjust parameters and iterate between the equations shown in Fig. 26.36. In general, for an ornithopter with equal mass, increasing the wingspan decreases the necessary flapping frequency. Alternatively, increasing the battery mass to improve flight time also requires increasing flapping frequency, electric power, and current to carry the extra payload. This explains why increasing battery mass beyond empty weight causes little increase in flight time, because the airframe needs to become much stronger at the cost of weight. A penalty in the flight time scaling equation needs to be implemented to correct for the increase in structural weight. The required flapping frequency and battery mass ratio specify the required power. Power increases significantly with wingspan. Additionally, power increases with added battery mass due to the increase in flapping frequency required to lift the larger mass. Finally, we can determine the current the battery needs to supply, which is proportional to the power assuming we use the same kind of battery and efficiency of motor. Iterating between these steps enables finding solutions that best meet the mission specifications. We note that many ornithopters could fly significantly longer by doubling their current battery mass (Fig. 26.36a) at the expense of control response (inertia) and airframe loading.

If flight time needs to increase for a wingspan-constrained ornithopter design, and battery mass and chemistry is already optimized, we should reduce air-

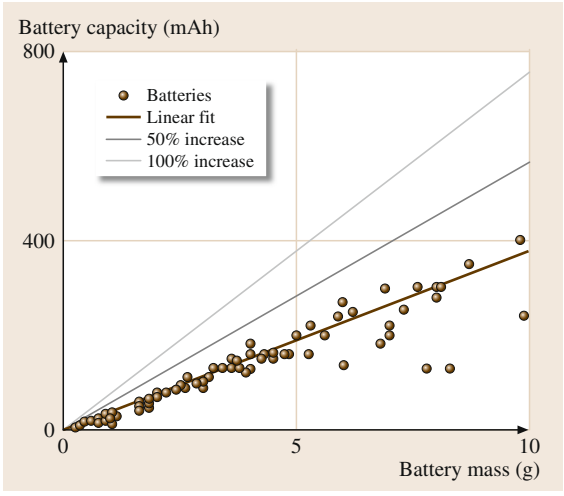


Fig. 26.35 Battery capacity as a function of mass for many small lithium polymer (LiPo) batteries in the size range (<10 g) which would be used for ornithopters with 10–50 cm wingspan. The graph shows the technology is linearly scalable. The approximate capacity density of small LiPo cells (3.7 V) is 37 mAh/g

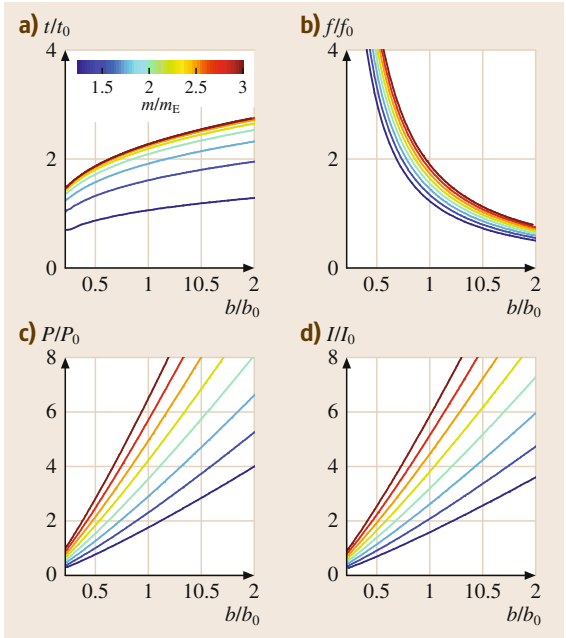


Fig.26.36a–d These four figures show the effects of changing wingspan and adding battery mass to an ornithopter on the flight time, power consumption, current requirement, and flapping frequency requirement. The value of the empty mass, m_e, is determined using the fitted curve in Fig. 26.34a for each wingspan. The figures are then scaled from the initial reference (Delfly II) whose position is at (1, 1) in each figure. **(a)** Increasing the battery mass ratio increases the flight time up until the ratio becomes equal to 3. This ignores additional airframe mass needed to carry these batteries. **(b)** However, increasing the battery mass also increases the required flapping frequency. **(c,d)** Increasing the frequency also increases the necessary power (P) and current (I). Using these parameters, we can iterate back and forth between the plots until a feasible design is found

Avidbots Ex1016
Page 695 of 2154

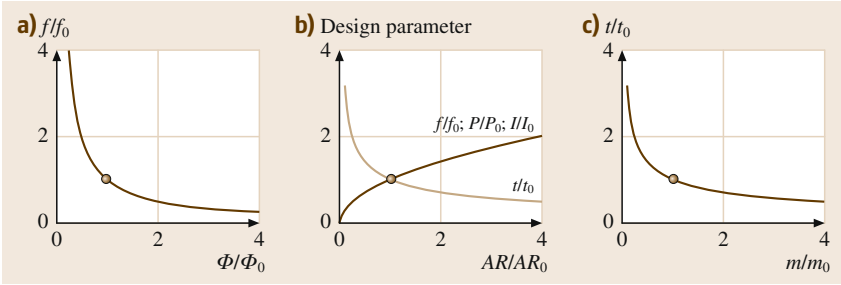


Fig.26.37a–c Changing additional parameters can modify performance of a scaled vehicle. **(a)** Adjusting the flapping amplitude allows the user to change the required flapping frequency to use available motor/gearbox combinations. Generally, larger flapping angles result in increased lift coefficient and decreased drag (after [26.59]). Thus, increasing the amplitude to match it with the motor and gear train can decrease the required power to fly. **(b)** As the aspect ratio increases at a constant wingspan, the wing area decreases, and therefore the flight time decreases while the required flapping frequency (and hence the power and current) increases. **(c)** Flight time decreases with additional payload (weight)

frame mass (Fig. 26.37) and increase wing area [26.49]. Mass can be further decreased by airframe optimization using underutilized aerospace optimization strategies, and by critically reevaluating the payload. Wing area can be increased by decreasing aspect ratio and selecting a biplane instead of a monoplane configuration. Whereas such wing design changes reduce aerodynamic efficiency of the wing, they increase the overall vehicle energy efficiency, and therefore increase flight time. Ornithopters that fly long enough to complete missions are often controlled by low-weight underpowered actuators that sacrifice maneuverability.

To control the ornithopter's flight and to utilize its maneuverability, we need to generate enough control torques with lightweight actuators. Designs optimized for flight time, such as the Delfly, use control surfaces added to the tail in the style of a traditional rudder or elevator. More maneuverable designs use the flapping wings as control surfaces, by changing their angle of attack (Nano-Hummingbird [26.51]) or left versus right wing relative flapping motions (Robobee [26.52]). The two dominant off the shelf actuators are standard servos and magnetic actuators. Standard servos have small electric motors and potentiometers and move to specified positions; while magnetic actuators have a small magnet inside a small coil of wire and apply specified amounts of torque. Magnetic actuators are available at lower masses than servos, which proves critical in optimizing performance of smaller ornithopters. This shows that selecting appropriate actuators involves a tradeoff between flight duration and maneuverability. Ornithopters that are more maneuverable require more powerful and precise servo actuators. The required servo torque of a scaled ornithopter can be estimated assuming isometric scaling: Torque should be proportional to total weight times wingspan, because aerodynamic force is proportional to weight, and arm length to wingspan. Knowing the required torque, we need to find a servo that can provide it. To reduce trial and error we have plotted current servo data to determine how torque correlates with mass to budget for its weight. The data in Fig. 26.38 shows that torque is proportional to mass squared for current servo technology,

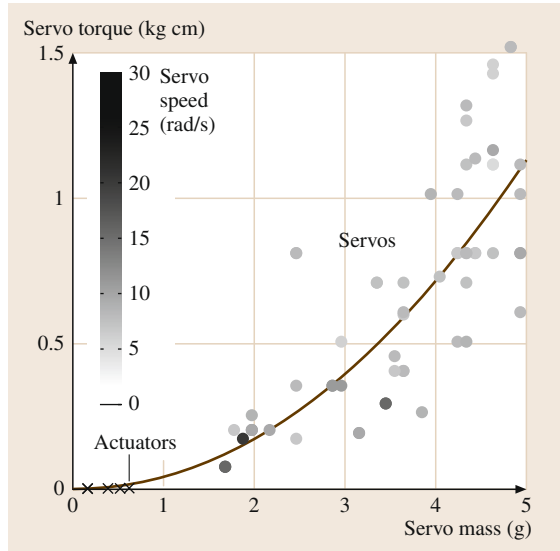


Fig. 26.38 Servo (*dots*) and actuator (*crosses*) torques increases with mass. The intensity of dots represents the servo speed, with darker dots representing faster servos (the magnetic actuators do not have speeds shown, as they apply a force rather than specify a position). The servo speed does not correlate strongly with mass, as it is dependent on the motors, gears, and other internal hardware of the servo, as well as the supply voltage. There are magnetic actuators available in the range of 0.8–1.8 g, they are not included here due to lack of data available from manufacturers

while empty ornithopter mass scales with wingspan to the power of 1.5 (Fig. 26.34), so as wingspan increases the actuator mass can become proportionally smaller.

We have demonstrated current design strategies based off scaling successful designs that ensure ornithopters fly. These upgraded *rules of thumb* are powerful because current aerospace design analysis and optimization techniques for ornithopters lack predictive power and are therefore less informative than estimates based on scaled flying designs. If current designers base their first iteration of new ornithopters on current state-of-the-art ornithopters, the field can progress at a faster pace through successful flight testing of new concepts that meet novel mission criteria.

26.7 System Integration and Realization

Enabling autonomous flights with UAS incorporates solving many challenges. This requires an interdisciplinary approach, bringing together expertise from many different fields. As shown in Fig. 26.39, knowledge in the field of aircraft design, as detailed in this chapter, is required, as well as in many fields of engineering and robotics.

26.7.1 Challenges for Autonomous UAS

Given the agility of UAS and their strict limitations on weight and power consumption, the choice of sensors, processors, and algorithms impose great technical and scientific challenges. Also, major differences exist between ground vehicles and UAS – sensors and algorithms that work well on ground vehicles cannot simply be applied on UAS due to inherent challenges.

Limited Payload and Power Supply

Weight and size restrictions require that lightweight sensors have to be used, usually at the cost of having noisier and less accurate data. These limitations also restrict the choice of onboard computers being used to process information from the sensors. Thus, algorithms need to cope with such data to achieve robust estimates, while having limited computational power. As a reference, the take-off weight of a commonly used small multirotor UAS (e.g., shown in Fig. 26.40) is 1.5 kg, including 0.5 kg of payload, while the flight autonomy is approximately 15 min. For this class of UAS, approximately 100 mW are required per 1 g additional take-off weight for hovering – a fact that has to be considered even for mounting small additional payload. Detailed studies of take-off weight, payload, and flight time of such systems can be found in [26.60].

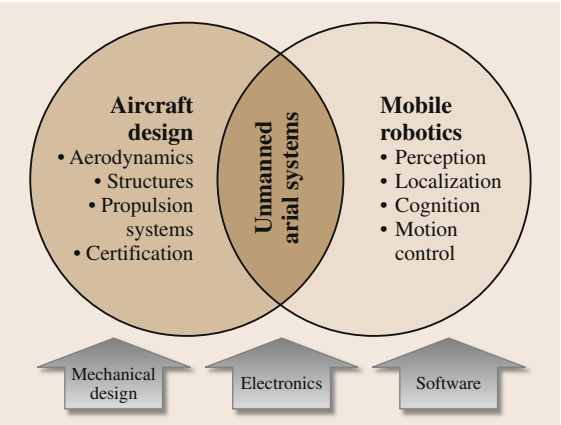


Fig. 26.39 UAS design and research is interdisciplinary as challenges from both aircraft design and mobile robotics are combined

Degrees of Freedom

Compared to typical ground vehicles, there are two additional degrees of freedom (DOF) for the vehicle's attitude (pitch and roll angle) as well as one additional degree of freedom for the altitude, that need to be estimated and controlled. This requires state estimation, control, and planning to be performed in full 6-D space, and without simplifying assumptions.

Under-Actuated Systems

The types of UAS studied here are usually underactuated, as there are less control inputs available than DOF. As a result, the attitude has to be changed for many maneuvers. This in return changes the field of view of onboard sensors interacting with the UAS' environment like cameras, or distance sensors.

Constant Motion and Inherent Instability

UAS cannot simply stop to acquire sensor readings, when state estimation is delayed or contains high uncertainty. While waiting for measurements or re-evaluation of uncertain state estimates, the vehicle continues moving and further falsifies these estimates.

26.7.2 Levels of Autonomy

We classify levels of autonomy and interaction with the pilot or operator into three categories: Manual flight, semiautonomous and autonomous. Industry and the re-

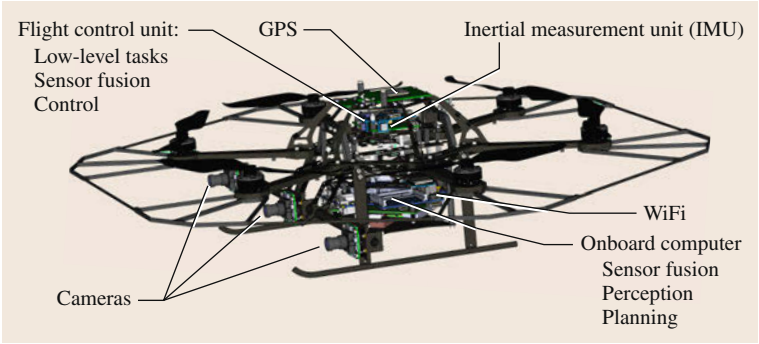


Fig. 26.40 Multi-rotor UAS and its main components (after [26.60])

search community have made great progress toward semiautonomous flights, while there are still many open questions and research topics for truly autonomous operation.

Manual Flight

We refer to this mode when operation of the UAS requires pilot skills. A remote-pilot has to handle attitude dynamics and throttle/thrust in a way such that the UAS remains in a stable state. That is, the pilot cannot leave the hands off the remote control. This also means that line-of-sight has to be maintained up to a distance where the pilot can observe the state of the UAS properly. Piloting may be aided by a stability augmentation system (SAS), like rate stabilization commonly used on fixed wing aircraft, or attitude control commonly used on multirotor UAS.

Semiautonomous (Automatic)

This flight mode does not require piloting skills anymore, which is why we refer to a pilot rather than an operator. Onboard sensors and algorithms are in charge of stabilizing the UAS, such that the operator can leave the hands off the remote control and the UAS remains in a stable state, waiting for input such as waypoints or desired velocities. However, it is still the full responsibility of the operator to find feasible paths, safely navigate around obstacles and to interact with other air traffic. This requires line of sight to be maintained, however, first-person view could be thought of, as long as legal requirements are fulfilled. Examples for such modes are off-the-shelf multirotor UAS [26.61] equipped with GPS sensors or small fixed-wing aircraft [26.62] used for surveying.

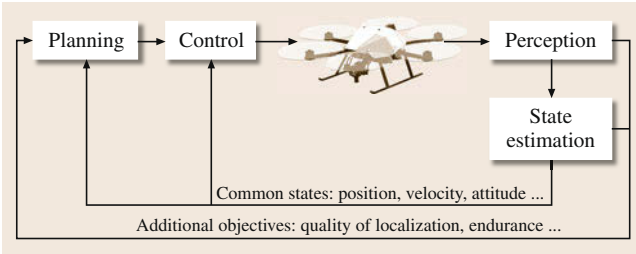


Fig. 26.41 Components for an autonomous UAS

Autonomous

Full autonomous mode relaxes the constraints from semiautonomous or automatic mode. Only mission goals are set, or high-level task allocation is handled by the operator, while the UAS navigates through the environment safely by itself. This includes global path planning, collision avoidance with both static and dynamic objects and replanning where necessary. The main idea is that the UAS can be *left alone* while performing the task at hand, and, in case, warns the operator ahead of time if intervention becomes necessary.

26.7.3 UAS Components

Autonomous UAS require numerous components from various fields, which need to be carefully designed in order to address the challenges detailed above. We highlight important design considerations of building blocks first, and show implementation aspects for real systems. Figure 26.40 shows the components for a commonly used multirotor UAS.

Components for Autonomous Flights

The levels of autonomy defined in Sect. 26.7.2 require several components to be designed and to work with each other. These components are shown in Fig. 26.41 and are described in the following:

Perception and State Estimation. Perception for autonomous UAS mainly involves both localization of the UAS and sensing of its environment. For localization, commonly used satellite-based localization systems such as GPS (GLONASS or GALILEO in the future) may not be accurate enough, especially during tasks that involve operation in close proximity to (man-made) structure, and certainly not indoors. Di-

Avidbots Ex1016
Page 698 of 2154

rect view to satellites is obstructed in these cases, compromising localization accuracy. Also, it may be considered whether sole reliance on these services is acceptable, as authorities may deteriorate the accuracy on purpose (*selected availability*). Thus, additional sensors such as cameras or laser-rangefinders should be used onboard UAS in combination with simultaneous localization and mapping (SLAM), or visual or laser odometry algorithms, yielding additional localization information [26.63–67].

Another important requirement to enable autonomous flights on the levels defined above, is sensing or reconstruction of the environment in order to navigate around obstacles, and for sense-and-avoid maneuvers with other air traffic. Obstacle sensing is especially important when navigating close to terrain at low altitude. Satellite-based navigation in combination with terrain maps may not be accurate enough, outdated, and cannot handle dynamic obstacles. Again, onboard sensors such as cameras, laser-rangefinders, or even radar, in combination with appropriate algorithms need to be used to overcome the limitations mentioned before [26.68, 69].

Having information from a multitude of sensors, from IMU over cameras to satellite-based, intelligent sensor fusion methods are required, taking all information into account and yielding a best estimate of the state of the UAS and its environment. As such, perception and state estimation are building blocks, providing essential information to the components described in the following.

Control. As UAS typically present highly dynamic uncertain flight behavior, appropriate and robust control techniques are required across all levels of autonomy. This ranges from SAS used for supporting the pilot during manual operation, to higher level control such as waypoint following, and more advanced techniques such as trajectory tracking, and may include handling of failure situations. Cascaded control structures are commonly used for this task [26.70]: inner- or low-level control loops stabilize attitude dynamics (these may be cascaded already), while outer- or high-level control loops stabilize translational dynamics (velocity or position). Employed control approaches range from simple P(I)D structures to more advanced techniques like model predictive control (MPC) [26.3]. During waypoint following, specified waypoints have to be reached and via-points have to be passed, while the velocity and attitude profile is left to the controller to optimize. For trajectory tracking in contrast, a tracking controller [26.71] has to follow specific profiles for position and attitude dynamics. These profiles are usually planned with approaches presented below, and yield more smooth paths than simple waypoint following.

Planning. Planning in the context of UAS depicts the process of planning appropriate waypoints or dynamic trajectories for a UAS depending on the mission at hand. This involves the latter two levels of autonomy described in the previous section. Requirements range from (dynamic) obstacle avoidance, taking obstacles and the dynamics of the UAS into account, to more sophisticated complete mission plans [26.72]. While taking into account vehicle dynamics is a straightforward choice, additional constraints such as quality of state estimation along a path, or battery endurance may have to be considered as well. Furthermore, there may be additional objectives like area covered or, for instance, energy optimizations for solar airplanes in order to stay airborne 24 h or longer.

While analytic approaches [26.73] can be used for simple planning tasks, random sampling-based planners [26.74–76] dominate the literature in the area of path planning for UAS, due to their ability to cope with nonlinear vehicle dynamics and high-dimensional state spaces. Based on these planners, a number of successful approaches exist for UAS, even taking into account localization uncertainty [26.4, 77–79].

Communication. Communication can be thought of the *glue* between the components mentioned above. While components have to communicate onboard across multiple computation devices, there is also the need of communication with a ground station, or with the operator remote control device, via radio-links. Here, the requirements in terms of range, delay, and transmission rate (commonly referred to as *bandwidth*) can vary greatly and should be tailored to the application: computationally heavy tasks, being offloaded to a ground station require high-bandwidth connections, where either WiFi (IEEE 802.11n or IEEE draft 802.11ac) or ultra wide band (UWB) techniques can be applied. However, a reasonable maximum range is in the order of 100 m. More autonomy means also less dependence on radio-links and realtime constraints. Missions involving larger distances have to rely on longer range radio-links, even up to satellite links, but at the expense of bandwidth and delay or significant weight and power requirements. That radio-links should never be used inside any (real-time) critical control loops, goes without saying.

Integration Onboard UAS

The components identified above have different requirements in terms of real-time constraints and computational complexity. Instead of real time, it is more

Avidbots Ex1016
Page 699 of 2154

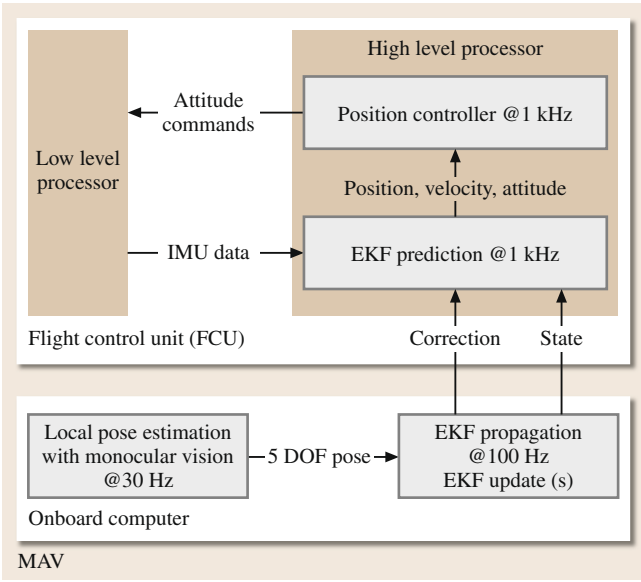


Fig. 26.42 Example implementation of flight-relevant components onboard a multirotor UAS. All time-critical tasks such as control and the prediction part of an extended Kalman filter (EKF) are performed on real-time microcontroller hardware, while computationally more demanding, but less time-critical, tasks (visual localization, EKF update) are processed on the onboard computer using a standard operating system

appropriate to consider that a task has to be completed *in time*. As an example, vehicle *control* tasks have to be computed within a few milliseconds, but these task are computationally less complex. *Path planning* in contrast is fairly complex, but it is usually perfectly fine if a path is computed within seconds, since planning horizons span much longer time. *Perception* and parts of *state estimation* lie in-between and need computation times in the order of tens of milliseconds. This defines how, and on which computation devices these components should be implemented. An example for a multirotor UAS is shown in Fig. 26.42. The microcontrollers on the flight-control-unit (FCU) are closest to inertial sensors and actuators, but least powerful. Control loops and the prediction parts of the state estimator are implemented here, running at guaranteed rates on a real-time operating system, or even without. This relaxes real-time constraints on the onboard computer, which can compute demanding tasks such as visual localization and update steps for state estimation. Local planning tasks are computed here as well, while less critical parts can be offloaded to a ground station.

Application Related Payload

While there is a set of sensors required to enable autonomous navigation, which are referred to as navigation sensors, additional sensors, or payload in general, need to be considered in the design phase. Not only in terms of payload, i. e., weight, but also such that the UAS is kept balanced. Furthermore, it has to be guaranteed that the additional payload does not interfere (electromagnetically) with components in the critical stabilization loop. Applications and their required sensors are detailed in the next section.

26.8 Applications of Aerial Robots

From an application perspective, one may distinguish between UAS that mostly operate as remote controlled or semiautonomous systems (typically referred to as drones) and intelligent systems, robots that present advanced levels of autonomy. Drones are essentially tele-operated aircraft or systems capable of tracking predefined trajectories while they further integrate onboard sensors to provide situational awareness. Most often such situational awareness is visual (using optics) but can also include meteorological and environmental tasks like hurricane-monitoring and chemical plume detection. As such, most drones fly high with a predetermined and structured flight plan and mission profile. Drones will continue to be valuable assets and as featured in daily media headlines world-wide, positively impact both civilian and military missions.

Nowadays, a constant trend is to develop aerial robots of advanced intelligence. Machine cognition, perception and vehicle control algorithms work in concert to perform applications that go beyond just situational awareness. Reaching new levels of autonomy, these robots are designed to handle unforeseen events, interact with their environment, and adapt to a broad range of scenarios. In essence, drones were in their vast majority *passive*, providing *eyes* on scene whereas modern aerial robots tend to become *active*, allowing their users to *engage* the scene, act *autonomously* and possibly *interact* with surroundings. Figure 26.43 presents indicative examples of current and emerging applications.

26.8.1 Demonstrated Applications of UAS

The Handbook's first edition listed eight categories of *possible* applications. Since then, all these applications have been realized, albeit with various levels of ma-

Avidbots Ex1016
Page 700 of 2154

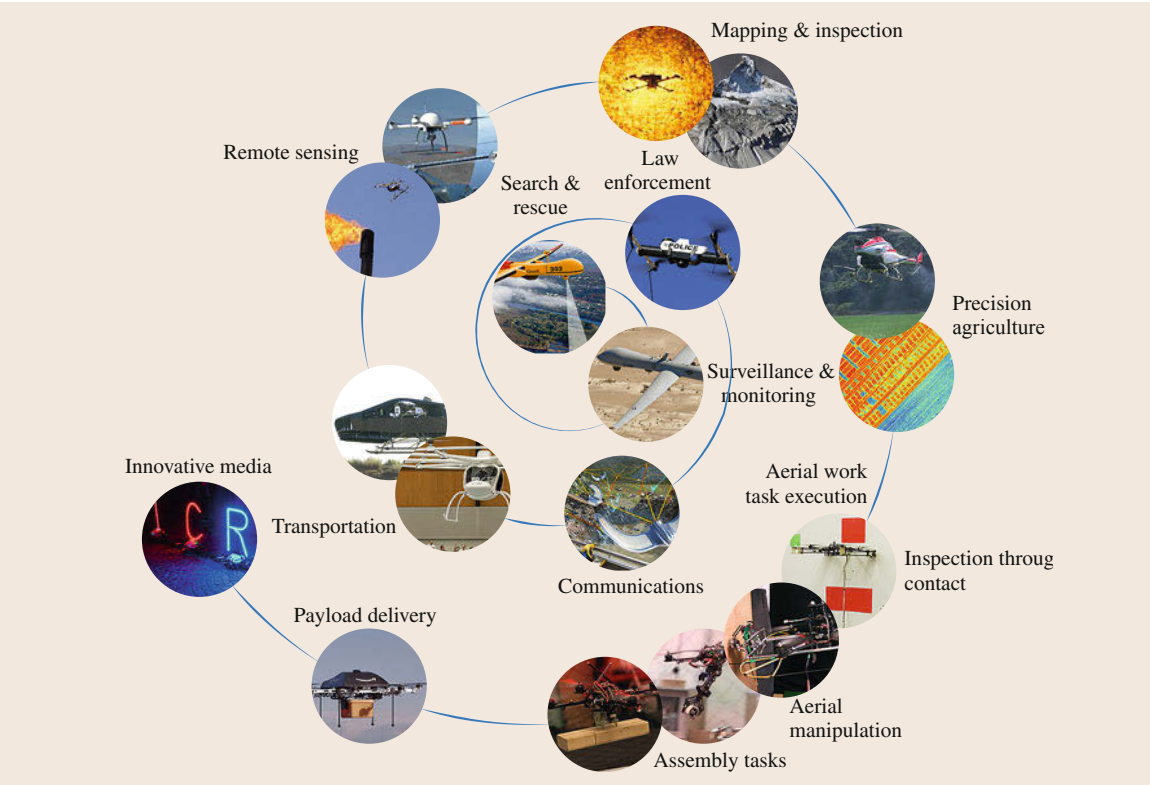


Fig. 26.43 Indicative demonstrated and emerging applications of aerial robotics

turity. UAS (both drones and aerial robots) have been deployed in:

1. Remote sensing
2. Disaster response
3. Surveillance
4. Search and rescue
5. Image acquisition
6. Communications
7. Transportation
8. And payload delivery.

The first five categories broadly fall into the area of reconnaissance, surveillance, and target acquisition (RSTA). Drones have continued to successfully perform RSTA-based tasks like volcanic sampling, damage assessment, border patrol, and cinematography. In contrast, the latter three categories are being realized with advanced aerial robots. Equipped with computational intelligence aerial robots can flock. Aerial robots can also airlift, maneuver and interact in near-Earth environments like in-and-around buildings, forests, caves and tunnels. As such current trends in aerial robotics push the application envelope beyond RSTA. The eight categories are now revisited with context on the expected breakthroughs to come while the topic of visual inspection for industrial purposes is separately discussed:

- *Remote sensing*: Drones are already used for pipeline spotting, power line monitoring, volcanic sampling, mapping, meteorology, geology, agriculture and unexploded mine detection. Advanced aerial robots will be able to conduct pipeline risk assessment and repair, power line maintenance, real-time mapping, crop care and mine defusing.
- *Disaster response*: Drones are also used for chemical sensing, flood monitoring, and wildfire management. Advanced aerial robots will be able to conduct infrastructure repair, flood mitigation, and wildfire fighting.
- *Surveillance*: Drones are employed for law enforcement, traffic monitoring, coastal and maritime patrol, and border patrols. Advanced aerial robots will be able to accomplish tasks like crowd control, traffic redirection, and inspection of maritime and trucking containers.
- *Search and rescue*: The vision of using drones for search and rescue operations within low-density or hard-to-reach areas has already become a reality.

Advanced aerial robots will be able to go beyond casualty extraction by assessing care and delivering first-aid support.

- *Visual inspection*: For quite some time, drones are utilized to provide direct visual feeds to ground operators inspection industrial and civil structures. Future aerial robots will be able to act as a new high fidelity autonomous inspection tools that could also be capable of conducting maintenance work tasks.
- *Transportation*: Drones are already used for small- and large-cargo transport, and possibly passenger transport. Advanced aerial robots will be able to conduct in-flight refueling of other aircraft, pick-up and drop-off of cargo as well as loading and extraction of casualties.
- *Communications*: The use of drones as permanent or ad hoc communication relays for voice and data transmission, as well as broadcast units for television or radio is currently a reality. Advanced aerial robots will be able to conduct perch-and-stare to serve as *bug on the wall* listening devices, perch-and-stare to harvest energy from power lines, flocking to establish networks especially in areas with degraded communications.
- *Payload delivery*: Drones are already employed for firefighting or crop dusting. Advanced aerial robots will be able to accomplish dexterous manipulation of payloads like tools for repairing structures, delivering, and insertion of logistics as well as handling crops.
- *Image acquisition*: Cinematography or real-time entertainment is another relatively established use-case of drones. Advanced aerial robots will be able to conduct acting as *pixels* and operate in formation flight to serve as physical displays in the air.

26.8.2 Current Applications and Missions

Since the Handbook's first edition, media headlines continue to document UAS and their impact in all the aforementioned application categories. No longer are such applications notional but are rather considered as routine. The following provides recent state-of-the-art examples in each category to underscore this message.

Drone usage for RSTA-based military operations is routine. Tracking targets and, in growing instances, destroying them, is commonly performed. But also non-military missions in remote sensing, disaster response, and surveillance are becoming routine. Beyond image acquisition, drones are routinely used to gather meteorological data. NASA and NOAA regularly deploy drones for real-time monitoring of hurricanes. Onboard sulfur dioxide sensors gather airborne samples from volcanic plumes. Drones are frequently deployed after disasters: in Fukushima (2011) and Hurricane Sandy (2012) aerial images were gathered to assess building damage. Monitoring of maritime piracy and cartel drug-trafficking also underscore the usefulness of drones to persistently survey using high-definition cameras and night vision systems.

Currently, drones are deployed to open and/or restricted areas where airborne collision risk is reduced. The need for situational awareness will however continue to be fulfilled by UAS. The demand for more data will push the development of newer drones. As such, well-known and cutting-edge aerodynamic design principles will be applied so that UAS fly longer and farther and incorporate different flight modalities like vertical take-off and landing (VTOL). Sensor suites will also grow in sophistication to collect higher-resolution and/or multispectral data. Propulsion technologies will also be engineered to enable UAS to carry more sensors. The net effect is that as missions become more routine and frequent, the variations of drones and their performance will grow to keep up with expanding RSTA-based needs.

26.8.3 Aerial Robots: Emerging Categories

The road ahead is full of challenging emerging applications that will benefit from the utilization of aerial robotic technologies. This is especially the case when one considers near-Earth environments like in-and-around buildings, through forests and down tunnels and caves. Beyond RSTA, such environments provide opportunities for aerial robots to dexterously interact with objects. Today's drones release retardants over wildfires but tomorrow's aerial robots would attach hoses and breach walls for firefighting. Interaction demands advances in areas like aircraft design (flight modalities, payload capability), algorithms (perception, control, motion planning, and grasping) and manipulators (arms and end-effectors). The latter two areas overlap with the greater domain of robotics research. As deeper understanding and realizations develop, the application space, demand and impact of aerial robots will grow even greater. In the following, a brief overview of exciting emerging applications will be listed.

Assembly Work Task Execution

One notional concept employs aerial robots to assembly physical structures. Such assembly pushes the boundaries of the categories of transportation and payload delivery. Labs like those of University of Pennsylvania [26.80], Switzerland (ETH Zurich [26.81]) have used gripper-mounted quadrotors to illustrate proof-of-concept. Demonstrations include the pickup and drop-off of workpieces to defined locations. These

workpieces self-connect using magnets or premachined joints but illustrate capabilities to coordinate multiple aircraft, mitigate disturbances like ground effect, sense and map, and execute agile maneuvers. Groups at Drexel, Twente, Seville, and Yale are equipping rotorcraft robots with manipulators to illustrate more dexterous tasks like inserting, screwing, and removing workpieces. Assembly and in general physical work-task execution will find applications in categories like disaster response (repairing structures) and remote sensing (inserting sensors).

Inspection Through Contact

Inspecting containers at docks, ships, and border crossings is a daunting task. Sensor technology is advancing to look inside such containers. However, dexterous aerial robots that can open hatches and trunks could also yield many applications. Inspection as an emerging category can employ aerial robots to physically interact with structures. Aerial robots can physically probe bridges, pipelines, and power lines to repair and replace parts. Interactive inspection could emerge to applications like crop handling and mine defusing. Labs like those of ETH Zurich [26.82] and the University of Bologna [26.83] have presented relevant results in the topic.

Innovative Media

Both the number of vendors and affordability of rotorcraft robots have increased since the Handbook's first edition. In mid-2000, it was not surprising to see a commercial quadrotor costing 40 000 USD. Today, there are dozens of companies providing robotic rotorcraft ranging in price (hundreds to under 10 000 USD), size (from pucks to bike wheels) and configurations (mono, co-axial, and multirotor). Such range will likely yield innovative and entrepreneurial applications.

The Firefly project at the MIT SENSEable Lab perhaps best illustrates the notion of coordinating multiple aerial robots to display images. Each aerial robot acts as flying *pixel* and maneuver into position to form 2 or 3-D images. Beyond visual and dynamic art, such capabilities could yield airborne displays for tasks like crowd control, traffic redirection and SOS signals. Airborne displays could also expand creative expression to enhance concerts, advertise at stadiums, and inform audiences.

Autonomous Structural Inspection and 3-D Reconstruction

From the perspective of the role of aerial robotics in accelerating growth, mapping of areas, structural inspection of buildings and infrastructure as well as recognition of objects or areas of interest is among the most important application fields. Aerial robots have achieved great milestones in such scenarios. Among others, recently researchers managed to extract a high-fidelity 3-D model of the Swiss mountain top Matterhorn utilizing a small FW-UAS and sophisticated robotic vision techniques [26.84], other efforts led to the inspection of a real power plant boiler using a multirotor vehicle and tightly integrated visual-inertial algorithms [26.85], while in terms of industrial adoption aerial robots have provenly minimized the inspection times of power infrastructure [26.86] and are often used in combination with geographic information system (GIS) data [26.84]. It is worth noting that some of the largest asset owners and service providers in the field of civil infrastructure and civil engineering works have shown interest and participate in large consortia that aim to make such aerial robotic technologies an integral and game-changing factor of how things are done (see e.g., Petrobot Project [26.87], ARCAS: Aerial Robotics Cooperative Assembly System [26.88], EuRoC: European Robotics Challenges [26.89], ARGOS Challenge [26.90].).

Precision Agriculture

Precision agriculture is considered to be among the fields where the use of aerial robots may become a key factor boosting growth and improving the production quality. The Japanese farming sector has historically been in a position to lead the relevant research efforts with successful designs like the Yamaha RMAX being extensively used for monitoring as well as spraying operations [26.5]. Already in December 2002, 1687 aerial robots were used in Japan to conduct precision agriculture operations. However, current research contributions and pioneering technology early adopters have opened an even more promising channel for the widest possible utilization of aerial robotic technologies for the benefit of agriculture production. Mainly through the use of miniaturized aerial robotics at farms of smaller scale and via the integration of advanced multispectral perception sensors (i. e., NDVI) [26.91], accurate mapping and analysis regarding the quality of the field, the level of plant growth and existence of diseases becomes possible. Such technologies are expected to become a critical tool to optimize and enhance agricultural services. Furthermore, manipulators-equipped aerial robots will be able to autonomously physically act based on their perception of the field and minimize the time and effort required for several agriculture services and tasks.

26.8.4 Open Issues

Emerging categories underscore both the distinction between more traditional drones and emerging aerial

robotic technologies as well as how much further the state-of-the-art has to be pushed. Clearly, today's UAS have limited payloads but adding manipulators goes beyond lift issues. When interacting with objects, aerial robots must handle reaction forces and torques. Understanding such reactions remains an open issue. However, both space and underwater roboticists have looked at dexterous manipulators for tasks like satellite repair and turning valves on ocean-based oil rigs. As research clarifies this issue and as lift capacities increase, dexterous aerial robots will be realized.

Airspace access is an often cited issue that may limit aerial robot development and vast utilization. A US congressional bill was passed in 2012 that sets requirements for UAS to fly in the national airspace by 2015. This has also sparked competition among a dozen or so states to establish UAS flight testing sites. In Europe, similar events have happened and there are UAS test sites in places like Finland (Kemijarvi), Sweden (NEAT), and Wales (Parc Aberporth).

Lastly, somewhat ironic is that today's unmanned drones require a crew of highly skilled operators. In the case of some Predator missions, crew sizes can be up to a dozen people. Also ironic is that human error is the most cited cause for drone accidents. As the number of UAS in the national airspace increases, the need for even more operators will also grow. This has the potential to raise the risk of UAS-related accidents. The issues of effective UAV pilot training, certifying operators, handling emergency landings, and sharing airports with manned aircraft will also emerge as pressing ones.

26.9 Conclusions and Further Reading

Design of aerial robots requires background knowledge in a multitude of subjects, from aerodynamics to dynamics, control, and system integration: we have overviewed the relevant basics along with analytical tools and guidelines to go through the stages of designing, modeling, and setting up operation of various types of unmanned aerial systems (UAS). An emphasis was given on custom tailoring a system to a specific application, in order to optimally meet related requirements in terms of endurance, range, agility, size, complexity, as well as from a system integration point of view. The compilation at hand shall serve as a starting point, further motivating the reader to study the various fields with their related literature, ranging from aircraft and system design to the classical autonomous robotics challenges involving perception, cognition and motion control.

Video-References

VIDEO 493 Delfly II in hover
available from http://handbookofrobotics.org/view-chapter/26/videodetails/493

VIDEO 602 AtlantikSolar field-trials
available from http://handbookofrobotics.org/view-chapter/26/videodetails/602

VIDEO 603 senseSoar UAV Avionics testing
available from http://handbookofrobotics.org/view-chapter/26/videodetails/603

VIDEO 604 Structural inspection path planning via iterative viewpoint resampling with application to aerial robotics
available from http://handbookofrobotics.org/view-chapter/26/videodetails/604

VIDEO 688 sFly: Visual-inertial SLAM for a small helicopter in large outdoor environments
available from http://handbookofrobotics.org/view-chapter/26/videodetails/688

VIDEO 689 UAV stabilization, mapping & obstacle avoidance using VI-sensor
available from http://handbookofrobotics.org/view-chapter/26/videodetails/689

VIDEO 690 Project Skye – autonomous blimp
available from http://handbookofrobotics.org/view-chapter/26/videodetails/690

VIDEO 693 Flight stability in aerial redundant manipulators
available from http://handbookofrobotics.org/view-chapter/26/videodetails/693

VIDEO 694 The astounding athletic power of quadcopters
available from http://handbookofrobotics.org/view-chapter/26/videodetails/694

VIDEO 695 Robots that fly ... and cooperate
available from http://handbookofrobotics.org/view-chapter/26/videodetails/695

VIDEO 696 A robot that flies like a bird
available from http://handbookofrobotics.org/view-chapter/26/videodetails/696

VIDEO 697 Robotic insects make first controlled flight
available from http://handbookofrobotics.org/view-chapter/26/videodetails/697

VIDEO 719 Towards valve turning using a dual-arm aerial manipulator
available from http://handbookofrobotics.org/view-chapter/26/videodetails/719

References

26.1 S. Leutenegger, M. Chli, R.Y. Siegwart: Brisk: Binary robust invariant scalable keypoints, Proc. IEEE Int. Conf. Comput. Vis. (ICCV) (2011) pp. 2548–2555

26.2 D. Scaramuzza, M.C. Achtelik, L. Doitsidis, F. Fraundorfer, E.B. Kosmatopoulos, A. Martinelli, M.W. Achtelik, M. Chli, S.A. Chatzichristofis, L. Kneip, G.H. Lee, S. Lynen, L. Meier, M. Pollefeys, A. Renzaglia, R. Siegwart, J.C. Stumpf, P. Tanskanen, C. Troiani, S. Weiss: Vision-controlled micro flying robots: From system design to autonomous navigation and mapping in GPS-denied environments, IEEE Robotics Autom. Mag. **2014**(9), 1–10 (2014)

26.3 K. Alexis, G. Nikolakopoulos, A. Tzes: Model predictive quadrotor control: Attitude, altitude and position experimental studies, IET Control Theory Appl. **6**(12), 1812–1827 (2012)

26.4 M.W. Achtelik, S. Lynen, S. Weiss, M. Chli, R. Siegwart: Motion and uncertainty aware path planning for micro aerial vehicles, J. Field Robotics **31**(4), 676–698 (2014)

26.5 K. Nonami: *Autonomous Flying Robots: Unmanned Aerial Vehicles and Micro Aerial Vehicles* (Springer, Berlin, Heidelberg 2010)

26.6 H. Tennekes: *The Simple Science of Flight: From Insects to Jumbo Jets* (MIT, Cambridge 2009)

26.7 W.J. Pisano, D.A. Lawrence: Control limitations of small unmanned aerial vehicles in turbulent environments, Proc. AIAA Guid. Navig. Control Conf. (2009)

26.8 B. Mettler, C. Dever, E. Feron: Scaling effects and dynamic characteristics of miniature rotorcraft, J. Guid. Control Dyn. **27**(3), 466–478 (2004)

26.9 ICAO: *Manual of the ICAO Standard Atmosphere: Extended to 80 Kilometres (262 500 feet)* (Int. Civil Aviation Organization (ICAO), Montréal 1993)

26.10 M. Hepparle: JavaFoil – Analysis of java airfoils, http://www.mh-aerotools.de/airfoils/javafoil.htm (2007)

26.11 M. Drela: XFOIL – Subsonic airfoil development system, http://www.web.mit.edu/drela/Public/web/xfoil (2000)

26.12 Techwinder: xlfr5, http://www.xflr5.com (2000)

26.13 B.W. McCormick: *Aerodynamics, Aeronautics, and Flight Mechanics* (Wiley, New York 1979)

26.14 G.J. Leishman: *Principles of Helicopter Aerodynamics* (Cambridge Univ. Press, Cambridge 2006)

26.15 G.D. Padfield: *Helicopter Flight Dynamics* (Blackwell, New York 2007)

26.16 L. Zaccarian: *Dc Motors: Dynamic Model and Control Techniques* (Lecture Notes Univ. Rome, Rome 2012)

26.17 M. Drela: AVL (Software for aerodynamic and flight-dynamic analysis), http://web.mit.edu/drela/Public/web/avl (2004)

26.18 B. Etkin: *Dynamics of Atmospheric Flight* (Wiley, New York 1972)

26.19 G.J.J. Ducard: *Fault-Tolerant Flight Control and Guidance Systems: Practical Methods for Small Unmanned Aerial Vehicles*, Advanced in Industrial Control (Springer, Berlin, Heidelberg 2009)

26.20 M.C.Y. Niu: *Airframe Structural Design* (Conmilit, Hong Kong 1988)

26.21 R. Randolph: *R/C Airplane Building Techniques*, R/C Encyclopedia (Air Age, Wilton 1991)

26.22 D.P. Raymer: *Aircraft Design: A Conceptual Approach* (AIAA, Washington 1989)

26.23 A. Noth: Design of Solar Powered Airplanes for Continuous Flight, Ph.D. Thesis (Ecole Polytechnique Fedenale de Lausanne, Lausanne 2008)

26.24 R.W. Beard, T.W. McLain: *Small Unmanned Aircraft: Theory and Practice* (Princeton Univ. Press, Princeton 2012)

26.25 A.A. Lambregts: *Vertical flight path and speed control autopilot design using total energy principles*, AIAA Paper (AIAA, Washington 1983)

26.26 S. Park, J. Deyst, J.P. How: A new nonlinear guidance logic for trajectory tracking, Proc. AIAA Guid. Navig. Control Conf. (2004) pp. 16–19

26.27 S. Bouabdallah, C. Bermes, S. Grzonka, C. Gimkiewicz, A. Brenzikofer, R. Hahn, D. Schafroth, G. Grisett, W. Burgard, R. Siegwart: Towards palm-size autonomous helicopters, Proc. Int. Conf. Exhib. Unmanned Aer. Veh. (2010)

26.28 P.I.E. Pounds, D.R. Bersak, A.M. Dollar: Grasping from the air: Hovering capture and load stability, Proc. IEEE Conf. Robotics Autom. (ICRA) (2011)

26.29 D. Gurdan, J. Stumpf, M. Achtelik, K.-M. Doth, G. Hirzinger, D. Rus: Energy-efficient autonomous four-rotor flying robot controlled at 1 khz, Proc. IEEE Int. Conf. Robotics Auton. Syst. (2007)

26.30 G.M. Hoffmann, H. Huang, S.L. Waslander, C.J. Tomlin: Quadrotor helicopter flight dynamics and control: Theory and experiment, Proc. AIAA Guid. Navig. Control Conf. (2007)

26.31 A. Ko, O.J. Ohnaian, P. Gelhausen: Ducted fan uav modeling and simulation in preliminary design, Proc. AIAA Model. Simul. Technol. Conf. Exhibit. (2007)

26.32 R. Naldi, F. Forte, L. Marconi: A class of modular aerial robots, Proc. 50th IEEE Conf. Decis. Control Eur. Control Conf. (2011)

26.33 E.R. Ulrich, J.S. Humbert, D.J. Pines: Pitch and heave control of robotic samara micro air vehicles, J. Aircr. **47**, 1290–1299 (2010)

Avidbots Ex1016
Page 705 of 2154

26.34 C.Y. Yun, I. Park, H.Y. Lee, J.S. Jung, I.S. Hwang, S.J. Kim: A new vtol uav cyclocopter with cycloidal blades system, Proc. 60th AHS Annu. Forum Amer. Helicopter Soc. (2004)

26.35 R.W. Prouty: *Helicopter Performance, Stability and Control* (Krieger, New York 2005)

26.36 M.B. Tischler, R.K. Remple: *Aircraft and Rotorcraft System Identification: Engineering Methods with Flight-Test Examples* (AIAA, Washington 2006)

26.37 B. Mettler: *Identification, Modeling and Characteristics of Miniature Rotorcraft* (Kluwer, Boston 2002)

26.38 A.R.S. Bramwell, G. Done, D. Balmford: *Bramwell's Helicopter Dynamics* (Butterworth-Heinemann, London 2001)

26.39 R.T.N. Chen: *Effects of primary rotor parameters on flapping dynamics*, Tech. Rep. (National Aeronautics and Space Administration, Washington 1980)

26.40 R.T.N. Chen: *A survey of nonuniform inflow models of rotorcraft flight dynamics and control applications*, Tech. Rep. (National Aeronautics and Space Administration, Washington 1989)

26.41 R. Cunha: Advanced Motion Control for Autonomous Air Vehicles, Ph.D. Thesis (Instituto Superior Tecnico, Universidade Tecnica de Lisbon, Lisbon 2007)

26.42 T.N. Pornsin-Sirirak, S.W. Lee, H. Nassef, J. Grasmeyer, Y.C. Tai, C.M. Ho, M. Keennon: MEMs wing technology for a battery powered ornithopter, Proc. 13th IEEE Annu. Int. Conf. MEMS (2000) pp. 799–804

26.43 T.N. Pornsin-Sirirak, Y.C. Tai, C.M. Ho, M. Keennon: Microbat: A palm-sized electrically powered ornithopter, Proc. NASA/SPL Workshop Biomorphic Robotics (2001) pp. 14–17

26.44 S.P. Sane: The aerodynamics of insect flight, J. Exp. Biol. **206**(23), 4191–4208 (2003)

26.45 M.H. Dickinson, F.O. Lehmann, S.P. Sane: Wing rotation and the aerodynamic basis of insect flight, Science **284**(5422), 1954–1960 (1999)

26.46 J. Young, S.M. Walker, R.J. Bomphrey, G.K. Taylor, A.L.R. Thomas: Details of insect wing design and deformation enhance aerodynamic function and flight efficiency, Science **325**(5947), 1549–1552 (2009)

26.47 L. Zhao, Q. Huang, X. Deng, S.P. Sane: Aerodynamic effects of flexibility in flapping wings, J. R. Soc. Interface **7**(44), 485–497 (2010)

26.48 F.O. Lehmann, S.P. Sane, M. Dickinson: The aerodynamic effects of wing-wing interaction in flapping insect wings, J. Exp. Biol. **208**(16), 3075–3092 (2005)

26.49 D. Lentink, S.R. Jongerius, N.L. Bradshaw: *Flying Insects and Robots* (Springer, Berlin, Heidelberg 2009)

26.50 G.C.H.E. de Croon, M.A. Groen, C. De Wagter, B. Remes, R. Ruijsink, B.W. van Oudheusden: Design, aerodynamics, and autonomy of the DelFly, Bioinspiration Biomim. **7(2)**, 025003 (2012)

26.51 M. Keennon, K. Klingebiel, H. Won, A. Andriukov: Tailless flapping wing propulsion and control development for the nano hummingbird micro air vehicle, Proc. Am. Helicopter Soc. Futur. Vert. Lift Aircr. Des. Conf. (2012)

26.52 K.Y. Ma, P. Chirarattananon, S.B. Fuller, R.J. Wood: Controlled flight of a biologically inspired, insect-scale robot, Science **340**(6132), 603–607 (2013)

26.53 D. Lentink, M.H. Dickinson: Rotational accelerations stabilize leading edge vortices on revolving fly wings, J. Exp. Biol. **212**(16), 2705–2719 (2009)

26.54 W. Shyy, H. Aono, C.-K. Kang, H. Liu: *An Introduction to Flapping Wing Aerodynamics* (Cambridge Univ. Press, Cambridge 2013)

26.55 C.P. Ellington, C. van den Berg, A.P. Willmott, A.L.R. Thomas: Leading-edge vortices in insect flight, Nature **384**, 626–630 (1996)

26.56 D.R. Warrick, B.W. Tobalske, D. Powers: Lift production in the hovering hummingbird, Proc. R. Soc. Biol. Sci. (2009) pp. 3747–3752

26.57 F.T. Muijres, L.C. Johansson, A. Hedenstrom: Leading edge vortex in a slow-flying passerine, Biol. Lett. **8**(4), 554–557 (2012)

26.58 J. Koo, T. Oka: *Experimental Study on the Ground Effect of a Model Helicopter Rotor in Hovering*, Tech. Rep. (NASA, Washington 1966)

26.59 S.P. Sane, M.H. Dickinson: The control of flight force by a applying wing: Lift and drag production, J. Exp. Biol. **204**(15), 2607–2626 (2001)

26.60 M.C. Achtelik, K.-M. Doth, D. Gurdan, J. Stumpf: Design of a multi rotor MAV with regard to efficiency, dynamics and redundancy, Proc. AIAA Guid. Navig. Control Conf. (2012)

26.61 Ascending Technologies Ltd.: http://www.asctec.de (2015)

26.62 Sensefly (Parrot Company): http://www.sensefly.com (2000)

26.63 M.W. Achtelik: Advanced Closed Loop Visual Navigation for Micro Aerial Vehicles, Ph.D. Thesis (ETH Zurich, Zurich 2014)

26.64 A. Bachrach, S. Prentice, R. He, N. Roy: RANGE – Robust autonomous navigation in GPS-denied environments, J. Field Robotics **28**, 644–666 (2011)

26.65 S. Leutenegger, P. Furgale, V. Rabaud, M. Chli, K. Konolige, R. Siegwart: Keyframe-based visual-inertial slam using nonlinear optimization, Proc. Robotics Sci. Syst. (RSS) (2013)

26.66 S. Weiss: Vision Based Navigation for Micro Helicopters, Ph.D. Thesis (ETH Zurich, Zurich 2012)

26.67 A.I. Mourikis, S.I. Roumeliotis, J.W. Burdick: SC-KF mobile robot localization: A stochastic cloning kalman filter for processing relative-state measurements, IEEE Trans. Robotics **23**(4), 717–730 (2007)

26.68 A. Bachrach, S. Prentice, R. He, P. Henry, A.S. Huang, M. Krainin, D. Maturana, D. Fox, N. Roy: Estimation, planning, and mapping for autonomous flight using an RGB-D camera in GPS-denied environments, Int. J. Robotics Res. **31**, 1320–1343 (2012)

26.69 T. Tomic, K. Schmid, P. Lutz, A. Domel, M. Kassecker, E. Mair, I. Grixa, F. Ruess, M. Suppa, D. Burschka: Toward a fully autonomous UAV: Research platform for indoor and outdoor urban search and rescue, IEEE Robotics Autom. Mag. **19**(3), 46–56 (2012)

26.70 S. Bouabdallah: Design and Control of Quadrotors with Application to Autonomous Flying, Ph.D. Thesis (STI School of Engineering, EPFL, Lausann 2007)

Avidbots Ex1016
Page 706 of 2154

26.71 T. Lee, M. Leoky, N.H. McClamroch: Geometric tracking control of a quadrotor UAV on SE(3), Proc. 49th IEEE Conf. Dec. Control (CDC) (2010) pp. 5420–5425

26.72 P. Doherty, J. Kvarnström, F. Heintz: A temporal logic-based planning and execution monitoring framework for unmanned aircraft systems, Auton. Agents Multi-Agent Syst. **19**(3), 332–377 (2009)

26.73 P.E. Hart, N.J. Nilsson, B. Raphael: A formal basis for the heuristic determination of minimum cost paths, Trans. Syst. Sci. Cybern. **4**(2), 100–107 (1968)

26.74 S. Karaman, E. Frazzoli: Incremental sampling-based algorithms for optimal motion planning, Proc. Robotics Sci. Syst. (RSS), Zaragoza (2010)

26.75 L.E. Kavraki, P. Švestka, J.-C. Latombe, M.H. Overmars: Probabilistic roadmaps for path planning in high-dimensional configuration spaces, IEEE Trans. Robotics Autom. **12**(4), 566–580 (1996)

26.76 S.M. LaValle, J.J. Kuffner: Randomized kinodynamic planning, Int. J. Robotics Res. **20**(5), 378–400 (2001)

26.77 A. Bry, N. Roy: Rapidly-exploring random belief trees for motion planning under uncertainty, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (2011) pp. 723–730

26.78 H. Cover, S. Choudhury, S. Scherer, S. Singh: Sparse tangential network (SPARTAN): Motion planning for micro aerial vehicles, IEEE Proc. Int. Conf. Robotics Autom. (ICRA) (2013)

26.79 R. He, S. Prentice, N. Roy: Planning in information space for a quadrotor helicopter in a gps-denied environments, IEEE Proc. Int. Conf. Robotics Autom. (ICRA) (2008) pp. 1814–1820

26.80 Q. Lindsey, D. Mellinger, V. Kumar: Construction with quadrotor teams, Auton. Robots **33**(3), 323–336 (2012)

26.81 J. Willmann, F. Augugliaro, T. Cadalbert, R. D'Andrea, F. Gramazio, M. Kohler: Aerial robotic construction towards a new field of architectural research, Int. J. Archt. Comput. **10**(3), 439–460 (2012)

26.82 G. Darivianakis, K. Alexis, M. Burri, R. Siegwart: Hybrid predictive control for aerial robotic physical interaction towards inspection operations, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (2014)

26.83 L. Marconi, R. Naldi, L. Gentili: Modelling and control of a flying robot interacting with the environment, Automatica **47**(12), 2571–2583 (2011)

26.84 Z. Lin: UAV for mapping – low altitude photogrammetric survey, Proc. 21st ISPRS Congr. Techn. Commis. I, Beijing (2008) pp. 1183–1186

26.85 J. Nikolic, M. Burri, J. Rehder, S. Leutenegger, C. Huerzeler, R. Siegwart: A UAV system for inspection of industrial facilities, Proc. IEEE Aerosp. Conf. (2013) pp. 1–8

26.86 Cyberhawk: Aerial Inspection and Suervying Specialists, http://www.thecyberhawk.com (2015)

26.87 Petrobot Project: http://www.petrobotproject.eu/

26.88 ARCAS: Aerial Robotics Cooperative Assembly System, http://www.arcas-project.eu/

26.89 EuRoC: European Robotics Challenges, http://www.euroc-project.eu/

26.90 ARGOS Challenge: http://www.argos-challenge.com/

26.91 E.R. Hunt Jr., W.D. Hively, S.J. Fujikawa, D.S. Linden, C.S.T. Daughtry, G.W. McCarty: Acquisition of nir-green-blue digital photographs from unmanned aircraft for crop monitoring, Remote Sens. **2**, 290–305 (2010)

Avidbots Ex1016
Page 707 of 2154

Multimedia Contents

27. Micro-/Nanorobots

Bradley J. Nelson, Lixin Dong, Fumihito Arai

The field of microrobotics covers the robotic manipulation of objects with dimensions in the millimeter to micron range as well as the design and fabrication of autonomous robotic agents that fall within this size range. Nanorobotics is defined in the same way only for dimensions smaller than a micron. With the ability to position and orient objects with micron- and nanometer-scale dimensions, manipulation at each of these scales is a promising way to enable the assembly of micro- and nanosystems, including micro- and nanorobots.

This chapter overviews the state of the art of both micro- and nanorobotics, outlines scaling effects, actuation, and sensing and fabrication at these scales, and focuses on micro- and nanorobotic manipulation systems and their application in microassembly, biotechnology, and the construction and characterization of micro and nanoelectromechanical systems (MEMS/NEMS). Material science, biotechnology, and micro- and nanoelectronics will also benefit from advances in these areas of robotics.

27.1 **Overview of Micro- and Nanorobotics** ... 671
27.2 **Scaling** ... 674
27.2.1 The Size of Things ... 674
27.2.2 Predominate Physics at the Micro- and Nanoscales ... 674
27.3 **Actuation at the Micro- and Nanoscales** 675
27.3.1 Electrostatics ... 675
27.3.2 Electromagnetics ... 676
27.3.3 Piezoelectrics ... 676
27.3.4 Other Techniques ... 676
27.4 **Imaging at the Micro- and Nanoscales** ... 676
27.4.1 Optical Microscopy ... 677
27.4.2 Electron Microscopy ... 677
27.4.3 Scanning Probe Microscopy ... 678
27.5 **Fabrication** ... 678
27.5.1 Microfabrication ... 679
27.5.2 Nanofabrication ... 680
27.6 **Microassembly** ... 681
27.6.1 Automated Microassembly Systems ... 681
27.6.2 Microassembly System Design ... 683
27.6.3 Basic Microassembly Techniques ... 685
27.7 **Microrobotics** ... 687
27.7.1 Introduction ... 687
27.7.2 Bio-Microrobotics ... 689
27.7.3 Bio-Mimetic/Untethered Microrobots ... 690
27.8 **Nanorobotics** ... 692
27.8.1 Introduction ... 692
27.8.2 Nanorobotic Manipulation Systems ... 695
27.8.3 Nanorobotic Manipulation and Assembly ... 696
27.8.4 Nanorobotic Systems ... 702
27.9 **Conclusions** ... 704
Video-References ... 704
References ... 705

27.1 Overview of Micro- and Nanorobotics

Progress in robotics over recent years has dramatically extended our ability to explore, perceive, understand, and manipulate the world on a variety of scales extending from the edges of the solar system, to the bottom

Avidbots Ex1016
Page 708 of 2154

of the sea, down to individual atoms (Fig. 27.1). At the lower end of this scale, technology has been moving toward greater control of the structure of matter, suggesting the feasibility of achieving thorough control of the molecular structure of matter atom by atom, as *Feynman* first proposed in 1959 in his prophetic article on miniaturization [27.1]:

> *What I want to talk about is the problem of manipulating and controlling things on a small scale... I am not afraid to consider the final question as to whether, ultimately – in the great future – we can arrange the atoms the way we want: the very atoms, all the way down!*

He asserted that

> *At the atomic level, we have new kinds of forces and new kinds of possibilities, new kinds of effects. The problems of manufacture and reproduction of materials will be quite different. The principles of physics, as far as I can see, do not speak against the possibility of maneuvering things atom by atom.*

This technology is now labeled *nanotechnology*.

The *great future* of Feynman began to be realized in the 1980s. Some of the capabilities he dreamed of have been demonstrated, while others are being actively pursued. Feynman foresaw the possibility of employing a microrobotic manipulator (*a master–slave system*) for bottom-up manufacturing (manipulation, assembly, etc.) of minute machines; one such device he described as *swallowing the surgeon*, which he attributed to his friend Albert R. Hibbs. He also imagined we could *build a billion tiny factories, models of each other, which are manufacturing simultaneously, drilling holes, stamping parts, and so on* [27.1]. Micro- and nanorobotics research has progressed from these seemingly *far-out* concepts of the 1960s and 1970s to reality when microelectromechanical systems (MEMSs) began to emerge in the late 1980s. These building blocks took the form of surface-micromachined micromotors and microgrippers made of polysilicon fabricated on a silicon chip [27.2]. In the late 1980s and early 1990s, more concrete suggestions on how one could realize MEMS-based microrobotic devices using such micromotors as well as potential applications were published [27.3, 4]. Today, a variety of microrobotic devices are enabling new applications in various fields.

In industry, interesting areas for microrobotics include assembly [27.5, 6], characterization, inspection and maintenance [27.7, 8], microoptics (positioning of microoptical chips, microlenses and prisms) [27.9], and microfactories [27.10]. Many of these applications require automated handling and assembly of small parts with accuracy in the submicron range.

Other important fields include biology (manipulation, capturing, sorting and combining cells [27.11]) and medical technology [27.12, 13]. In surgery, the use of steerable catheters and endoscopes is very attractive and the development of increasingly small microrobotic devices is rapidly progressing. Wireless untethered microrobots that will explore and repair our bodies (*swallowing the surgeon*) appear to be simply a matter of time. In fact, endoscopy using wireless capsules (camera pills) are already on the market and allow for endoscopic imaging of the entire gastrointestinal tract [27.14], something currently not possible using standard scopes. Magnetic steering or crawling-type motions serve as promising ways for such devices to locomote in a controlled fashion [27.15]. Doctors could steer pill-mounted cameras and other actuators to areas of interest for visual investigation and biopsies beyond the range of current endoscopes.

Nanorobotics represents the next stage in miniaturization for maneuvering nanoscale objects. Nanorobotics is the study of robotics at the nanometer scale, and includes robots that are nanoscale in size, i. e., nanorobots, and large robots capable of manipulating objects that have nanometer dimensions with nanometer resolution, i. e., nanorobotic manipulators. The field of nanorobotics brings together several disciplines, including nanofabrication processes used for producing nanoscale robots, nanoactuators, nanosensors, and physical modeling at nanoscales. Nanorobotic manipulation technologies, including the assembly of nanometer-sized parts, the manipulation of biological cells or molecules, and the types of robots used to perform these types of tasks also form a component of nanorobotics.

As the 21st century unfolds, the impact of nanotechnology on the health, wealth, and security of humankind is expected to be at least as significant as the combined influences in the 20th century of antibiotics, the integrated circuit, and human-made polymers. For example, N. Lane stated in 1998, [27.16]

> *If I were asked for an area of science and engineering that will most likely produce the breakthroughs of tomorrow, I would point to nanoscale science and engineering.*

The great scientific and technological opportunities nanotechnology presents have stimulated extensive exploration of the nanoworld and initiated an exciting worldwide competition, which has been accelerated by the publication of the *National Nanotechnology Initiative* by the US government in 2000 [27.17]. Nanorobotics will play a significant role as an enabling nanotechnology and could ultimately be a core part of

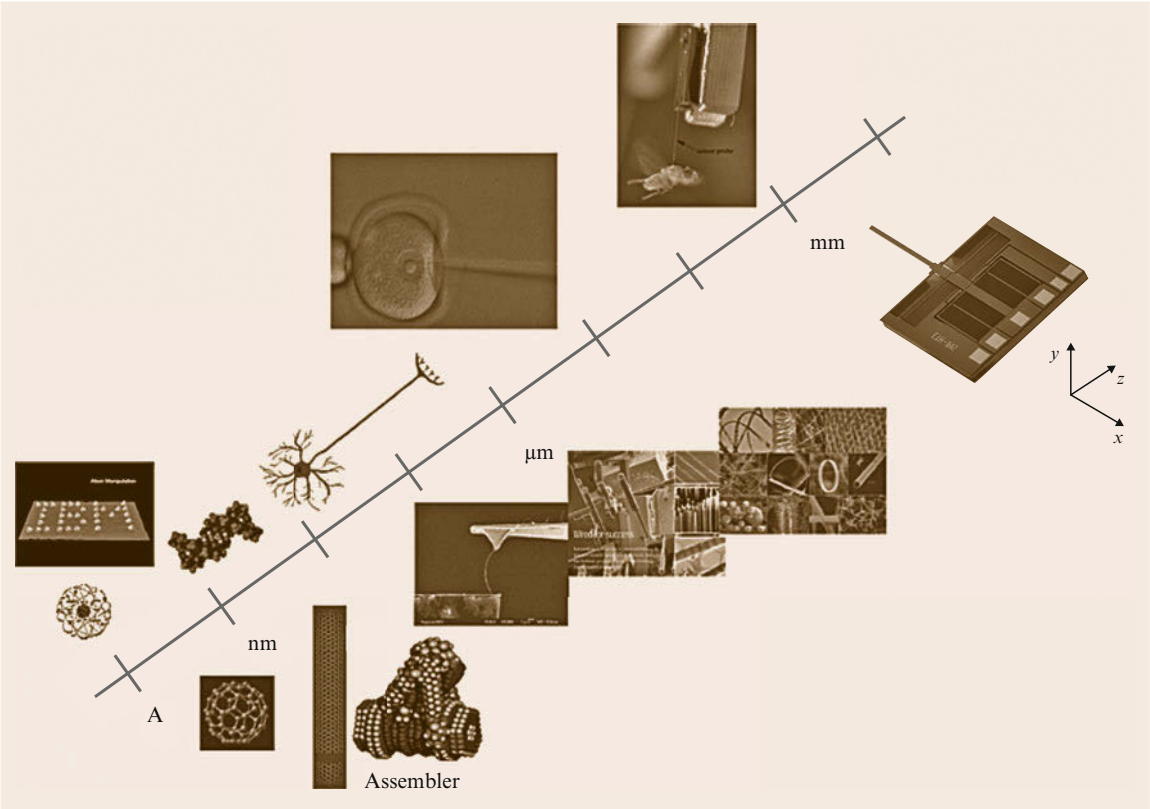


Fig. 27.1 Robotic exploration at micro- and nanometer scales

nanotechnology if *Drexler*'s machine-phase nanosystems based on self-replicative molecular assemblers via mechanosynthesis can be realized [27.18].

By the early 1980s, scanning tunneling microscopes (STMs) [27.19] radically changed the way in which we interacted with and even regarded single atoms and molecules. The very nature of proximal probe methods encourages the exploration of the nanoworld beyond conventional microscopic imaging. Scanning probes now allow us to perform *engineering* operations on single molecules, atoms, and bonds, thereby providing a tool that operates at the ultimate limits of fabrication. They have also enabled exploration of molecular properties on an individual nonstatistical basis.

STMs and other nanomanipulators are nonmolecular machines that use bottom-up strategies. Although performing only one molecular reaction at a time is obviously impractical for making large amounts of a product, it is a promising way to provide the next generation of nanomanipulators. Most important, it is possible to realize the directed assembly of molecules or supermolecules to build larger nanostructures through nanomanipulation. The products produced by nanomanipulation could be the first step of a bottom-up strategy in which these assembled products are used to self-assemble into nanomachines.

One of the most important applications of nanorobotic manipulation will be nanorobotic assembly. However, it appears that, until assemblers capable of replication can be built, the combination of chemical synthesis and self-assembly are necessary when starting from atoms; groups of molecules can self-assemble quickly due to their thermal motion, enabling them to *explore* their environments and find (and bind to) complementary molecules. Given their key role in natural molecular machines, proteins are obvious candidates for early work in self-assembling artificial molecular systems. *Degrado* [27.20] demonstrated the feasibility of designing protein chains that predictably fold into solid molecular objects. Progress is also being made in artificial enzymes and other relatively small molecules that perform functions like those of natural proteins. Several bottom-up strategies using self-assembly appear feasible [27.21]. Chemical synthesis, self-assembly, and supramolecular chemistry make it possible to provide building blocks at relatively large sizes beginning from the nanometer scale. Nanorobotic manipulation serves as the base for a hybrid approach to

Avidbots Ex1016
Page 710 of 2154

construct nanodevices by structuring these materials to obtain building blocks and assembling them into more complex systems.

Despite the claims of many *futurists*, the form nanorobots of the future will take and what tasks they will actually perform remain unclear. However, it is clear that nanotechnology is progressing towards the construction of intelligent sensors, actuators, and systems that are smaller than 100 nm. These nanoelectromechanical systems (NEMSs) will serve as both the tools to be used for fabricating future nanorobots as well as the components from which these nanorobots may be developed. Shrinking device size to these dimensions presents many fascinating opportunities such as manipulating nano-objects with nanotools, measuring mass in femtogram ranges, sensing forces at piconewton scales, and inducing gigahertz motion, among other new possibilities waiting to be discovered. These capabilities will, of course, drive the tasks that future nanorobots constructed by and with NEMS will perform. NEMS and the components of nanorobots will be the products of nanorobotic manipulation. Large nanorobotic manipulators will be able to shrink in size due to this development, thus enabling nanosized robotic manipulators and other forms of nanorobots. All of these form the scope of the area of nanorobotics.

This chapter focuses on micro- and nanorobotics including actuation, manipulation, and assembly at the micro- and nanoscale. The main goal of these fields of robotics is to provide an effective technology for the experimental exploration of the micro- and nanoworld, and to push the boundaries of this exploration from a robotics research perspective.

27.2 Scaling

27.2.1 The Size of Things

Things we can potentially observable range from 10^{-35} m (the Planck length) to 10^{26} m (the radius of the observable universe). A nanometer, 10^{-9} m, is about ten times the size of the smallest atoms, such as hydrogen and carbon, while a micron is barely larger than the average wavelength of visible light, thus invisible to the human eye. A millimeter, the size of a pinhead, is roughly the smallest part typically fabricated using traditional machining techniques. The range of scales from millimeters to nanometers is one million (Fig. 27.1), which is also about the range of scales in present-day mechanical technology from the largest skyscrapers to the smallest conventional mechanical machine parts. The vast opportunity to make new machines spanning almost six orders of magnitude from 1 mm to 1 nm, is one take on *Feynman*'s famous statement, *there is plenty of room at the bottom* [27.1]. If L is taken as a typical length, 0.1 nm for an atom, perhaps 2 m for a human, this scale range in L would be 2×10^{10}. If the same scale range were to apply to an area, 0.1 nm × 0.1 nm versus 2 m × 2 m, the scale range for area L^2 is 4×10^{20}. Since a volume L^3 is enclosed by sides L, we can see that the number of atoms of size 0.1 nm in a ($2\,m^3$) volume is about 8×10^{30}, recalling that Avogadro's number $N_A = 6.022 \times 10^{23}$ is the number of atoms in a gram-mole, supposing that the atoms were ^{12}C, molar responding to a density $1.99 \times 10^4\,kg/m^3$. A primary working tool of the nanotechnologist is facility in scaling the magnitudes of various properties of interest, as the length scale L shrinks, e.g., from 1 mm to 1 nm.

Clearly, the number of atoms in a device scales as L^3. If a transistor on the micron scale contains 10^{12} atoms, then on the nanometer scale, $L'/L = 10^{-3}$ it will contain 1000 atoms, likely too few to preserve its function.

Normally, we will think of scaling as an isotropic scale reduction in three dimensions. However, scaling can be thought of usefully when applied only to one or two dimensions, scaling a cube to a two-dimensional (2-D) sheet of thickness a or to a one-dimensional (1-D) tube or nanowire of cross-sectional area a^2. The term *zero-dimensional* (0-D) is used to describe an object small in all three dimensions, having volume a^3. In electronics, a zero-dimensional object (a nanometer-sized cube a^3 of semiconductor) is called a quantum dot (QD) or *artificial atom* because its electronic states are few and sharply separated in energy, and thus resemble the electronic states of an atom.

27.2.2 Predominate Physics at the Micro- and Nanoscales

The predominate physics at the micro- and nanoscale can be dramatically different from at the macroscale. Surface and intermolecular forces, such as adhesion forces originating from surface tension forces, van der Waals forces, and electrostatic forces, become more significant than volumetric forces such as gravitational forces for objects with sizes well below 1000 μm [27.22]. Although the laws of classical Newtonian physics may well suffice to describe changes in behavior down to 10 nm (100 Å), the range of scaling

is tremendous. Therefore, the changes in magnitudes of many important physical properties, such as resonant frequencies, are so great, that completely different applications may appear.

The more challenging question for the nanotechnologist is to understand and hopefully to exploit those changes in physical behavior that occur at the end of the classical scaling range. The *end of the scaling* is the size scale of atoms and molecules, where nanophysics [27.23] is the proven conceptual replacement of the laws of classical physics. Modern physics, which includes quantum mechanics as a description of matter on a nanometer scale, is a very well-developed and proven subject whose application to real situations is limited only by modeling and computational competence.

In the modern era, simulations and approximate solutions increasingly facilitate the application of nanophysics to almost any problem of interest. Many central problems are already (adequately, or more than adequately) solved in the extensive literatures of theoretical chemistry, biophysics, condensed matter physics, and semiconductor device physics. The practical problem is to find the relevant work, and, frequently, to convert the notation and units systems to apply the results to the problem at hand.

27.3 Actuation at the Micro- and Nanoscales

The positioning of nanorobots and nanorobotic manipulators depends largely on nanoactuators. While nanosized actuators for nanorobots are still under exploration and relatively far from implementation, MEMS-based efforts are focused on shrinking their sizes [27.24]. Nanometer resolution motion has been extensively investigated and can be generated using various actuation principles. Electrostatics, electromagnetics, and piezoelectrics are the most common ways to realize actuation at nanoscales. For nanorobotic manipulation, besides nanoresolution and compact sizes, actuators generating large strokes and high forces are best suited for such applications. The speed criteria are of less importance as long as the actuation speed is in the range of a couple of hertz and above. Table 27.1 provides a small selection of early works on actuators [27.25–30] suitable for micro- and nanorobotic applications (partially adapted from [27.24]).

Several extensive reviews on various actuation principles have been published [27.4, 31–34]. During the design of an actuator, the trade-offs among range of motion, force, speed (actuation frequency), power consumption, control accuracy, system reliability, robustness, load capacity, etc. must be taken into consideration. This section reviews basic actuation technologies and potential applications at nanometer scales.

27.3.1 Electrostatics

Electrostatic charge arises from a build up or deficit of free electrons in a material, which can exert an attractive force on oppositely charged objects, or a repulsive force on similarly charged objects. Since electrostatic fields arise and disappear rapidly, such devices will likewise demonstrate very fast operation speeds and be little affected by ambient temperatures.

Previous investigations have produced many examples of miniature devices using electrostatic force for actuation including silicon micromotors [27.35, 36], microvalves [27.37], and microtweezers [27.38]. This type of actuation is important for achieving nanoscale actuation.

Electrostatic fields can exert great forces, but generally across very short distances. When the electric field must act over larger distances, a higher voltage will be required to maintain a given force. The extremely low current consumption associated with electrostatic devices makes for highly efficient actuation.

Table 27.1 Actuation with MEMS

Actuation principle	Type of motion	Volume (mm^3)	Speed (s^{-1})	Force (N)	Stroke (m)	Resolution (m)	Power density (W/m^3)	Reference
Electrostatic	Linear	400	5000	1×10^{-7}	6×10^{-6}	NA	200	[27.25]
Magnetic	Linear	$0.4 \times 0.4 \times 0.5$	1000	2.6×10^{-6}	1×10^{-4}	NA	3000	[27.26]
Piezoelectric	Linear	$25.4 \times 12.7 \times 1.6$	4000	350	1×10^{-3}	7×10^{-8}	NA	[27.27]
Actuation principle	**Type of motion**	**Volume (mm^3)**	**Speed (rad/s)**	**Torque (Nm)**	**Stroke (rad)**	**Resolution (rad)**	**Power density (W/m^3)**	**Reference**
Electrostatic	Rotational	$\pi/4 \times 0.5^2 \times 3$	40	2×10^{-7}	2π	NA	900	[27.28]
Magnetic	Rotational	$2 \times 3.7 \times 0.5$	150	1×10^{-6}	2π	$5/36\pi$	3000	[27.29]
Piezoelectric	Rotational	$\pi/4 \times 1.5^2 \times 0.5$	30	2×10^{-11}	0.7	NA	NA	[27.30]

Avidbots Ex1016
Page 712 of 2154

Table 27.2 Comparison of nano actuators

Method	Efficiency	Speed	Power density
Electrostatic	Very high	Fast	Low
Electromagnetic	High	Fast	High
Piezoelectric	Very high	Fast	High
Thermomechanical	Very high	Medium	Medium
Phase change	Very high	Medium	High
Shape memory	Low	Medium	Very high
Magnetostrictive	Medium	Fast	Very high
Electrorheological	Medium	Medium	Medium
Electrohydrodynamic	Medium	Medium	Low
Diamagnetism	High	Fast	High

27.3.2 Electromagnetics

Electromagnetism arises from electric current moving through a conducting material. Attractive or repulsive forces are generated adjacent to the conductor and proportional to the current flow. Structures can be built which gather and focus electromagnetic forces, and harness these forces to create motion.

Electromagnetic fields arise and disappear rapidly, thus permitting devices with very fast operation speeds. Since electromagnetic fields can exist over a wide range of temperatures, performance is primarily limited by the properties of the materials used in constructing the actuator.

One example of a microfabricated electromagnetic actuator is a microvalve which uses a small electromagnetic coil wrapped around a silicon micromachined valve structure [27.39]. The downward scalability of electromagnetic actuators into the micro- and nanorealm may be limited by the difficulty of fabricating small electromagnetic coils. Furthermore, most electromagnetic devices require perpendicularity between the current conductor and the moving element, presenting a difficulty for planar fabrication techniques commonly used to make silicon devices.

An important advantage of electromagnetic devices is their high efficiency in converting electrical energy into mechanical work. This translates into less current consumption from the power source.

27.3.3 Piezoelectrics

Piezoelectric motion arises from the dimensional changes generated in certain crystalline materials when subjected to an electric field or to an electric charge. Structures can be built which gather and focus the force of the dimensional changes, and harness them to create motion. Typical piezoelectric materials include quartz (SiO_2), lead zirconate titanate (PZT), lithium niobate, and polymers such as polyvinylidene fluoride (PVDF).

Piezoelectric materials respond very quickly to changes in voltages and with great repeatability. They can be used to generate precise motions with repeatable oscillations, as in quartz timing crystals used in many electronic devices. Piezoelectric materials can also act as sensors, converting tension or compression strains to voltages.

On the microscale, piezoelectric materials have been used in linear inchworm drive devices and micropumps [27.40]. STMs and most nanomanipulators use piezoelectric actuators.

Piezoelectric materials operate with high force and speed, and return to a neutral position when unpowered. They exhibit very small strokes (under 1 %). Alternating electric currents produce oscillations in the piezoelectric material, and operation at the sample's fundamental resonant frequency produces the largest elongation and highest power efficiency [27.41]. Piezoelectric actuators working in the *stick–slip* mode can provide millimeter to centimeter strokes. Most commercially available nanomanipulators adopt this type of actuators, such as Picomotors from New Focus and Nanomotors from Klock.

27.3.4 Other Techniques

Other techniques include thermomechanical, phase change, shape memory, magnetostrictive, electrorheological, electrohydrodynamic, diamagnetism, magnetohydrodynamic, shape changing, polymers, and biological methods (living tissues, muscle cells, etc.). Table 27.2 compares these techniques.

27.4 Imaging at the Micro- and Nanoscales

A brief overview is given of common imaging tools used in the research area of micro- and nanorobotics including optical, electron, and scanning probe microscopy. The application and integration of these tools for micro- and nanorobotics are discussed in separate sections.

In selecting a proper tool for imaging/sensing at these scales, the following factors should be first considered:

1. Specimen: size, conductivity, and environment compatibility are among the most important aspects

Avidbots Ex1016
Page 713 of 2154

Table 27.3 Comparison of optical microscopes (OM) and electron microscopes

Feature	OM	TEM	SEM
General use	Surface morphology and sections (1–40 μm)	Section (40–150 nm) or small particles on thin membranes	Surface morphology
Source of illumination	Visible light	High-speed electrons	High-speed electrons
Best resolution	ca. 200 nm	ca. 0.2 nm	ca. 3–6 nm
Magnification range	10–1000×	500–500.000×	20–150.000×
Depth of field	0.002–0.05 nm (N.A. 1.5)	0.004–0.006 mm (N.A. 10^{-3})	0.003–1 mm
Lens type	Glass	Electromagnetic	Electromagnetic
Image ray-formation spot	On eye by lenses	On phosphorescent plate by lenses	On cathode tube by scanning device

to consider. For example, in vivo bioapplications generally require air or liquid, so lower-resolution, light microscopy should be the first choice. If higher resolution is needed, atomic force microscopy or scanning near-field optical microscopy can be used.

2. Resolution: the ability to see fine details of a specimen. Once you can resolve fine details then you can magnify them. Every microscope has a finite resolution; if you magnify objects beyond the resolution the result will be empty magnification. Roughly speaking, a light microscope cannot provide a resolution better than 200 nm. The best commercially available scanning electron microscopes (SEM) have approximately 1 nm resolution, while transmission electron microscopes (TEM) can achieve approximately 0.2 nm, and a scanning tunneling microscope (STM) working in ultrahigh vacuum under very low temperature can resolve atomic-level structures. A comparison of optical microscopy and electron microscopy is given in Table 27.3.
3. Depth of field: the range of depth that a specimen is in acceptable focus. A microscope that has a small depth of field will have to be continuously focused up and down to view a thick specimen.
4. Contrast: the ratio between dark and light. Typically, most microscopes use absorption contrast, i. e., the specimen is subjected to stains in order to be seen. This is called bright-field microscopy. There are other types of microscope that use more exotic means to generate contrast, such as phase contrast, dark field, and differential interference contrast.
5. Brightness: the amount of light. The higher a microscope magnifies the more light will be required. The illumination source should also be at a wavelength (color) that will facilitate interaction with the specimen. All microscopes fall into either of two categories based on how the specimen is illuminated. In the typical compound microscope the light passes through the specimen and is collected by the image forming optics. This is called diascopic illumination. Dissecting (stereo) microscopes generally use episcopic illumination for use with opaque specimen. The light is reflected onto the specimen and then into the objective lens.

27.4.1 Optical Microscopy

Since their invention in the late 1500s, light microscopes have enhanced our knowledge in basic biology, biomedical research, medical diagnostics, and materials science. Light microscopes can magnify objects up to 1000 times, revealing microscopic details. Light-microscopy technology has evolved far beyond the first microscopes of Robert Hooke and Antoni van Leeuwenhoek. Special techniques and optics have been developed to reveal the structures and biochemistry of living cells. Most optical microscopes in current use are known as compound microscopes, where a magnified image of an object is produced by the objective lens, and this image is magnified by a second lens system (the ocular or eyepiece) for viewing. Microscopes have even entered the digital age, using charge-coupled devices (CCDs) and digital cameras to capture images.

The development of modern microscopy has enabled a large family of optical microscopes. For special purposes, other types of optical microscopes can be selected. These include phase-contrast microscopy, fluorescence microscopy, confocal scanning optical microscopy, and deconvolution microscopy image reconstruction.

27.4.2 Electron Microscopy

Scanning Electron Microscope (SEM)

Since the commercial availability of the SEM in 1966, it has been a valuable resource for viewing samples at a much higher resolution and depth of field than the typical optical microscope. Conventional SEMs can resolve down to the nanometer scale (≈ 1 nm) whereas an optical microscope can only resolve down to approximately 200 nm [27.42]. Unlike conventional optical microscopes, SEMs have a high depth of field which gives imaged samples a three-dimensional appearance.

Early SEMs were limited to viewing conductive samples. However, many of today's SEMs can image nonconductive samples in addition to conductive samples using variable-pressure chambers.

Transmission Electron Microscope (TEM)

The TEM can resolve to an atomic scale down to about 1 d.h. (i.e., 0.1 nm). The TEM mode of operation is similar to that of the SEM in that both microscopes contain an electron gun source of illumination. However, the TEM detects the electrons that pass through a given sample. As a result, the electron gun of the TEM operates at higher energy levels between 50–1000 kV, while the SEM's electron gun operates at around 1–30 kV. In order for proper imaging to take place, the sample must be very thin so that electrons from the beam can pass through the specimen. Electrons that do not pass through the sample cannot be detected. Unlike the SEM, the TEM produces images that are two dimensional in appearance.

27.4.3 Scanning Probe Microscopy

Scanning Tunneling Microscope (STM)

Similar to the TEM, the STM [27.19] can also resolve specimens down to the atomic scale. The scanning probe of the STM is comprised of a noble metal sharpened to an atomic-sized tip, which is mounted on a piezoelectrically driven (x, y, z) linear stage. The STM makes use of the quantum-mechanical effect known as tunneling. Electron tunneling occurs when electrons, driven by a small potential difference, flow across the gap between the probe tip and sample. This event takes place at Ångström-scale distances between the probe tip and the sample [27.43]. The tunneling current, which is typically on the order of a few nanoamperes, is directly related to the tip–sample separation distance. Thus, the tunneling current can be measured and is kept at a constant value by controlling the tip–sample gap distance (z) with a feedback control system. The probe tip is then scanned (x, y) along the entire surface of the sample. Since the control system maintains a constant tunneling current, and thus maintains a constant tip–sample distance (z), the result of a scan yields a $z(x, y)$ terrain map of the sample with enough resolution to detect atomic-scale features. The STM can achieve faster imaging by operating in so-called constant-height mode in which the probe tip is scanned in a plane parallel to the average surface portion. The tip–sample distance (z) can then be inferred directly from the measured tunneling current [27.43].

Atomic Force Microscope (AFM)

The AFM [27.44] is considered to be a spin-off of the STM. One shortcoming of the STM is that it requires conductive probe tips and samples to work properly. The AFM was developed in order to view nonconductive samples, giving it a wider applicability than the STM. In addition to imaging nonconductive samples, the AFM can also image samples immersed in liquid, which is useful for biological applications [27.45]. Although the STM and AFM are similar in that they both scan the surface of a sample with an atomically sharp probe, they operate under slightly different principles. The AFM is based on interatomic forces as opposed to the electron tunneling used in the STM. The AFM probe tip is mounted on the end of a microscale cantilever beam. At very short separations, the forces between atoms in the probe tip and atoms in the sample cause the cantilever to deflect. This deflection is usually measured by striking the back of the cantilever with a laser. The reflection of the laser beam hits a photodetector, which can be used to recover the deflection of the cantilever. The force can then be calculated by using Hooke's law, which simply relates the applied force to the stiffness and deflection of a material. The forces that are measured can be on the scale of piconewtons [27.43]. The AFM has three main modes of operation known as contact mode, noncontact mode, and tapping mode.

Unlike the SEM and TEM, both the STM and AFM do not require a vacuum environment in order to function. However, a high vacuum is advantageous in order to keep the samples from becoming contaminated from the surrounding environment as well as controlling humidity. In addition, atomic resolution in air is hardly possible with an AFM due to humidity. As a result of humidity, a water film is formed and creates problems because of capillary forces. This can be resolved by operating in vacuum environment or completely in a liquid solution [27.43].

27.5 Fabrication

The design of micro- and nanorobotic devices is inextricably linked to available fabrication techniques. However, though the development of microfabrication processes has become somewhat stable over the past decade, nanofabrication processes are still being actively pursued, and the design constraints generated by these processes are relatively unexplored. This section will briefly highlight the processes used in conventional

microfabrication including lithography, thin-film deposition, chemical etching, and electrodeposition, and describe some emerging techniques for nanofabrication.

Most micro- and nanofabrication techniques have their roots in the standard fabrication methods developed for the semiconductor industry [27.46–48]. Therefore, a clear understanding of these techniques is necessary for anyone embarking on a research and development path in the micro/nano area.

27.5.1 Microfabrication

In this section, we will discuss the major microfabrication techniques used most frequently in the manufacturing of microstructures.

Photolithography

Lithography is the technique used to transfer a computer-generated pattern onto a substrate (silicon, glass, GaAs, etc.). This pattern is subsequently used to etch an underlying thin film (oxide, nitride, etc.) for various purposes (doping, etching, etc.). Although photolithography, i. e., lithography using an ultraviolet (UV) light source, is by far the most widely used lithography technique in microelectronic fabrication, electron-beam (e-beam) and X-ray lithography are two alternatives that have attracted considerable attention in the MEMS and nanofabrication areas. We will discuss photolithography in this section and postpone the discussion of e-beam and X-ray techniques to the subsequent sections dealing with nanofabrication.

The starting point following the creation of the computer layout for a specific fabrication sequence is the generation of a photomask. This involves a sequence of photographic processes (using optical or e-beam pattern generators) that results in a glass plate having the desired pattern in the form of a thin (≈ 100 nm) chromium layer. Following the generation of photomask, the lithography and etching process can proceed as shown in Fig. 27.2. After depositing the desired material on the substrate, the photolithography process starts with spin-coating the substrate with a photoresist. This is a polymeric photosensitive material that can be spun onto the wafer in liquid form; usually an adhesion promoter such as hexamethyldisilazane (HMDS) is used prior to the application of the resist. The spinning speed and photoresist viscosity will determine the final resist thickness, which is typically 0.5–2.5 µm. Two different kinds of photoresist are available: positive and negative. With a positive resist, the UV-exposed areas will be dissolved in the subsequent development stage, whereas with a negative photoresist, the exposed areas will remain intact after the development. After spinning the photoresist on the wafer, the substrate is soft-baked (5–30 min at 60–100 °C in order to remove the solvents from the resist and improve the adhesion. Subsequently, the mask is aligned to the wafer and the photoresist is exposed to a UV source.

After exposure, the photoresist is developed in a process similar to the development of photographic film. The resist is subsequently hard-baked (20–30 min at 120–180 °C in order to further improve the adhesion. The hard-bake step concludes the photolithography sequence by creating the desired pattern on the wafer. Next, the underlying thin film is etched, and the photoresist is stripped in acetone or other organic solvent. Figure 27.3 shows a schematic of the photolithography steps with a positive photoresist.

Thin-Film Deposition and Doping

Thin-film deposition and doping are used extensively in micro- and nanofabrication technologies. Most of the fabricated structures contain materials other than that of the substrate, which are obtained by various deposition techniques, or by modification of the substrate. These techniques include oxidation, doping, chemical vapor deposition (CVD), physical vapor deposition (PVD), and electroplating.

Etching and Substrate Removal

For micro- and nanofabrication, in addition to thin film etching, often the substrate (silicon, glass, GaAs, etc.) needs to be removed in order to create various mechanical structures (beams, plates, etc.). Two important figures of merit for any etching process are selectivity and directionality.

Selectivity is the degree to which the etchant can differentiate between the masking layer and the layer to

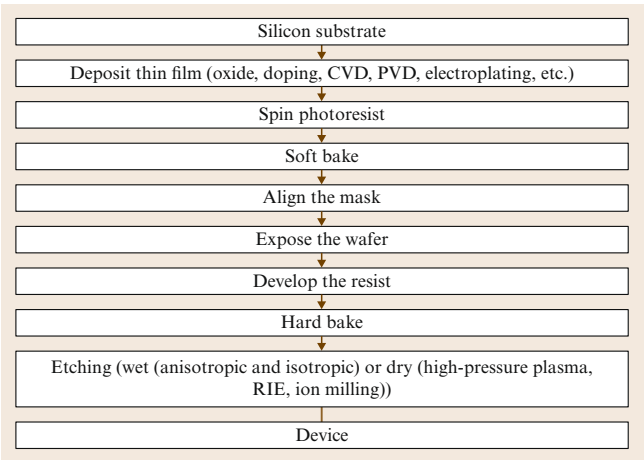


Fig. 27.2 Typical microfabrication process flow

Avidbots Ex1016
Page 716 of 2154

be etched. Directionality has to do with the etch profile under the mask. In an isotropic etch, the etchant attacks the material in all directions at the same rate, creating a semicircular profile under the mask (Fig. 27.4a). In an anisotropic etch, the dissolution rate depends on specific directions, and one can obtain straight sidewalls or other noncircular profiles (Fig. 27.4b). One can also divide the various etching techniques into wet and dry categories. Due to the lateral undercut, the minimum feature size achievable with wet etchants is limited to $> 3\,\mu m$. Photoresist and silicon nitride are the two most common masking materials for the wet oxide etch. Anisotropic and isotropic wet etching of crystalline (silicon and gallium arsenide) and noncrystalline (glass) substrates are important topics in micro- and nanofabrication [27.49–53].

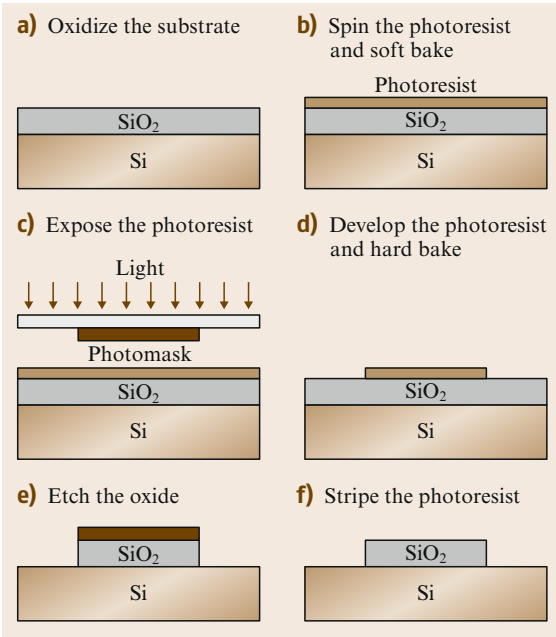


Fig.27.3a–f Schematic drawing of the photolithographic steps with a positive photoresist (PR) **(a)** Oxidize the substrat **(b)** spin the photoresist and soft bake **(c)** expose the photoresist **(d)** develop the photoresist and hard bake **(e)** etch the oxide **(f)** stripe the photoresist

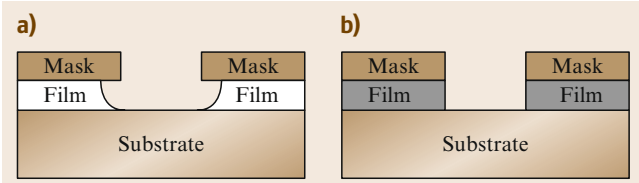


Fig.27.4a,b Profile for isotropic **(a)** and anisotropic **(b)** etch through a photoresist mask

The anisotropic behavior of these etchants with respect to the (111) planes have been used extensively to create beams, membranes, and other mechanical and structural components. Figure 27.5 shows the typical cross sections of (100) silicon wafers etched with an anisotropic wet etchant. As can be seen, the (111) slow planes are exposed and creating 54.7° sloped sidewalls. Depending on the dimensions of the mask opening, a V-groove or a trapezoidal trench is formed in the (100) wafer. A large enough opening will allow the silicon to be etched all the way through the wafer, thus creating a thin dielectric membrane of the other side. It should be mentioned that exposed convex corners have a higher etch rate than the concave ones, resulting in an undercut that can be used to create dielectric (e.g., nitride) cantilever beams.

Dry etching techniques are largely plasma based. They have several advantages compared with wet etching. These include smaller undercut (allowing smaller lines to be patterned) and higher anisotropy (allowing high-aspect-ratio vertical structures). However, the selectivity of dry etching techniques is lower than the wet etchants, and one must take into account the finite etch rate of the masking materials. The three basic dry etching techniques, namely high-pressure plasma etching, reactive-ion etching (RIE), and ion milling, utilize different mechanisms to obtain directionality.

27.5.2 Nanofabrication

The design and fabrication of NEMS is an emerging area being pursued by an increasing number of researchers. Two approaches to nanofabrication, top-down and bottom-up, have been identified by the nanotechnology research community and are being independently investigated by various researchers. Top-down approaches are based on microfabrication and include technologies such as nanolithography, nanoimprinting, and chemical etching. Presently, these are 2-D fabrication processes with relatively low resolution. Bottom-up strategies are assembly-based techniques. Currently these strategies include techniques such as self-assembly, dip-pen lithography, and directed self-assembly. These techniques can generate regular nanopatterns at large scales.

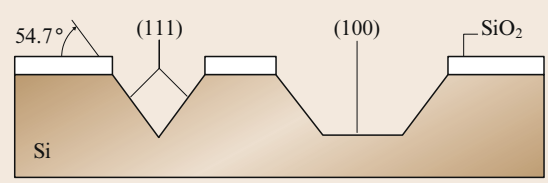


Fig. 27.5 Anisotropic etch profiles for (100) silicon wafers

Avidbots Ex1016
Page 717 of 2154

In this section, we will discuss three major nanofabrication techniques. These include:

1. e-beam and *nanoimprint* fabrication
2. *Epitaxy* and *strain engineering*, and
3. Dip-pen nanolithography.

E-beam Lithography and Nanoimprint Fabrication

In previous sections, we discussed several important lithography techniques commonly used in MEMS and microfabrication. These include various forms of UV (regular, deep, and extreme) and X-ray lithography. However, due to the lack of resolution (in the case of the UV), or the difficulty in manufacturing mask and radiation sources (X-ray), these techniques are not suitable for nanometer-scale fabrication. E-beam lithography is an attractive alternative technique for fabricating nanostructures [27.53]. It uses an electron beam to expose an electron-sensitive resist such as polymethyl methacrylate (PMMA) dissolved in trichlorobenzene (positive) or polychloromethylstyrene (negative).

The e-beam gun is usually part of an SEM, although a TEM can also be used. Although electron wavelengths on the order of 1 Å can easily be achieved, electron scattering in the resist limits the attainable resolutions to > 10 nm. The beam control and pattern generation are achieved through a computer interface.

E-beam lithography is serial and hence has a low throughput. Although this is not a major concern in fabricating devices used in studying fundamental microphysics, it severely limits large-scale nanofabrication. E-beam lithography, in conjunction with such processes as lift-off, etching, and electrodeposition, can be used to fabricate various nanostructures.

An interesting new technique that circumvents the serial and low-throughput limitations of the e-beam lithography for fabricating nanostructures is nanoimprint technology [27.54]. This technique uses an e-beam-fabricated hard material master (or mold) to stamp and deform a polymeric resist. This is usually followed by a reactive-ion etching step to transfer the stamped pattern to the substrate. This technique is economically superior, since a single stamp can be used repeatedly to fabricate a large number of nanostructures.

Epitaxy and Strain Engineering

Atomic-precision deposition techniques such as molecular-beam epitaxy (MBE) and metallo-organic chemical vapor deposition (MOCVD) have proven to be effective tools in fabricating a variety of quantum confinement structures and devices (quantum well lasers, photodetectors, resonant tunneling diodes, etc.) [27.55–57].

Dip-Pen Nanolithography

In dip-pen nanolithography (DPN), the tip of an AFM operated in air is *inked* with a chemical of interest and brought into contact with a surface. The ink molecules flow from the tip onto the surface as with a fountain pen. Line widths down to 12 nm with spatial resolution of 5 nm have been demonstrated with this technique [27.58]. Species patterned with DPN include conducting polymers, gold, dendrimers, deoxyribonucleic acid (DNA), organic dyes, antibodies, and alkanethiols.

27.6 Microassembly

Assembly is often required in macroscale product manufacturing in order to reduce the complexity and cost of the manufacturing process. Assembly makes it possible to build complex products from relatively simple parts and to integrate incompatible manufacturing processes. It also makes maintenance and replacement possible. The extension of assembly techniques into the microscale is driven by the development of modern design and manufacturing technologies in the pursuit of miniaturization and function integration, especially by the development of integrated circuit (IC) [27.59] and MEMS fabrication techniques. With the extension of manufacturing technology into the microscale and even nanoscale domain, the term microassembly has been created to refer specifically to assembly operations performed at the micro/mesoscale [27.60]. To give a formal definition, microassembly is the assembly of objects with microscale and/or mesoscale features under microscale tolerances [27.61].

27.6.1 Automated Microassembly Systems

Microassembly plays the role of an enabling technology in various processes of MEMS fabrication, including device fabrication, packaging, and interconnection. MEMS device fabrication is fundamentally different from the highly modular IC fabrication in that it often requires the machining of complex-shaped three-dimensional (3-D) mechanical structures [27.51]. However, almost all current MEMS fabrication tech-

niques are subject to constraints in limited allowable materials, limited capabilities in true 3-D fabrication, and the requirement of fabrication process compatibility. Microassembly provides a possible solution to these constraints. For example, incompatible fabrication processes can be integrated through assembly. This makes it possible to use *nontraditional* fabrication techniques that are not necessarily based on semiconductor materials, such as laser cutting, microwire electrical discharge machining (EDM), and micromilling [27.62]. Complex 3-D structures can also be developed using parts with relatively simple geometry [27.63, 64]. Microassembly is also crucial to MEMS packaging and interconnection [27.65, 66].

From the perspective of robotic systems and automation, MEMS device fabrication and packaging share many common assembly requirements. A fundamental commonality of both processes is the requirement of the ability to manipulate micro/mesoscale objects so that precise (i. e., microscale tolerance) spatial relations can be established (e.g., die alignment, part insertion) and certain physical/chemical processes (e.g., die bonding, surface coating) can be performed. Another common requirement is to control the interaction force involved. MEMS devices often have fragile structures such as thin beams or membranes. This requires controlling the interactive force in manipulation operations. Typically, the force magnitude resides in the range from millinewtons to micronewtons.

MEMS devices are often three-dimensional. MEMS packaging requires both electrical interconnection for signal transmission and mechanical interconnection for the interaction of the packaged device with its external environment [27.51, 67]. Many such mechanical interconnections require three-dimensional manipulation and three-dimensional force control. The actual operations are highly application specific, which pose great challenges to the development of automated microassembly systems. Automated IC packaging systems can be used for the packaging of certain MEMS devices such as accelerometers and gyros. However, packaging of microfluidic devices, optical MEMS devices, and hybrid microsystems often require the development of new automated microassembly techniques and systems.

The selection of assembly mode is among the first decisions to be made in developing automated microassembly systems. Sequential microassembly requires the use of micromanipulators and sensory feedback. At each moment, only one or a few parts are being assembled. Depending on the physical effects used, parallel microassembly can be either deterministic or stochastic [27.68]. Die bonding is an example of deterministic parallel microassembly. In stochastic parallel microassembly, large numbers of parts are assembled simultaneously using distributed physical effects such as electrostatic force, capillary force, centrifugal force, or vibration [27.60, 69–71]. In fact, the basic philosophy of stochastic parallel microassembly is to minimize the use of sensory feedback.

Each of these assembly modes has both advantages and disadvantages. Each has its own suitable applications. Deterministic parallel assembly shares several commonalities with sequential microassembly. For example, sensory feedback is often used in deterministic parallel microassembly. However, deterministic parallel microassembly requires high relative positioning accuracy between parts. In addition, only simple planar structure features can be assembled in order to make parallel operation possible.

Due to the requirement of MEMS packaging for three-dimensional manipulation and microassembly, it can be expected that automated sequential microassembly will be the most widely adopted solution. In particular, those MEMS packaging applications that require the control of multiple-degree-of-freedom (DOF) interaction force should use sequential microassembly. The major possible disadvantage of sequential microassembly is its low throughput. This constraint can often be overcome by appropriate system design.

Here we introduce a three-dimensional microassembly example that originates from an industrial application [27.61]. It is significantly different from wire bonding and die bonding applications in that high-precision 3-D part insertion is required. Together with other examples, it will be used throughout this section to illustrate the major concepts and techniques and the logic connections between each functional unit of an automated microassembly system.

The assembly task is to pick up micromachined thin metal parts that are transferred to the assembly workcell on a vacuum-release tray (Fig. 27.6a–c), and insert them into vertically deep reactive ion etching (DRIE) etched holes in a silicon wafer (Fig. 27.6d). The wafers can have diameters up to 8 inches. The holes on each wafer form regular arrays consisting of approximately 50 holes each. However, these arrays may not be regularly distributed on the wafer. Typically, hundreds of parts are to be assembled on each wafer. In general, each assembly operation is a typical rectangular-peg-into-a-rectangular-hole problem. Each metal part is approximately half a millimeter in width and less than 100 μm in thickness at its rectangular tip. The total assembly tolerance is typically smaller than 10 μm in the vertical direction, and smaller than 20 μm in the horizontal direction. This task is a typical application of 3-D microassembly techniques.

27.6.2 Microassembly System Design

This section discusses the design of automated microassembly systems from the perspective of robotic systems and automation. Performance objects to be achieved include high reliability, high throughput, high flexibility, and low cost.

General Guidelines

Taking a System Perspective. An automated microassembly system consists of many functional units and must integrate techniques from a diverse range of areas such as robotics, computer vision, microscope optics, physics, and chemistry. It is therefore important to consider the interaction between these units.

Emphasizing the Coupling with Packaging Processes. Although in this section microassembly techniques are addressed primarily from the perspective of robotics for the clarity of presentation, robotic system designers must recognize the strong dependence of architecture design of the robotic systems on the packaging process being implemented. This connection should be emphasized from the beginning of system development.

Design for Reconfigurability. Automated packaging machines must be designed so that they can be customized for a wide variety of applications. Reconfigurability is therefore a basic design requirement. Typically, modular design based on function decomposition is desirable. Support for tool replacement is essential.

The design of automated microassembly systems is strongly dependent on the assembly tolerance required. First of all, the repeatability of the motion control system and the micromanipulator used is determined by the required assembly tolerance. In addition, assembly tolerance often determines the minimum resolution of the microscope optics. General microassembly tasks may only require microscopic vision feedback and manipulators with microscale repeatability, while complex microassembly tasks may also require the integration of microforce and vision feedback [27.72, 73].

Automated microassembly systems must be able to support a wide variety of material-handling tools, which include part transfer tools, bulk feeders, wafer-handling tools, magazine loaders and unloaders, etc.

The role of the micromanipulator is to provide multiple-DOF fine motion control. The role of the microgripper is to grasp objects in pick-and-place and other assembly operations. Their efficiency and robustness will to a great extent decide the performance of the entire system. Microgripper design is also closely related to the design of fixtures for microassembly operations.

Major environment factors include clean-room requirements, temperature, humidity, airflow, etc. Certain assembly operations must be performed in a clean room. This requires that the design of automated microassembly systems comply with relevant standards. Some packaging processes such as eutectic bonding must be performed under high temperature. Consequently, the potential influence of high temperature on motion control system and microscope optics must be considered. For the manipulation of microscale objects, environment conditions such as temperature and humidity can have a major influence on adhesion forces [27.74, 75]. Therefore, it is often important to consider environment control in system design.

Assembly Process Flow: An Example

The micromachined metal parts (Fig. 27.6c) are horizontally transferred to the workcell on the vacuum-release tray (Fig. 27.6b). The wafer is placed on a wafer mount perpendicular to the horizontal plane (Fig. 27.6b). This configuration does not require the flipping of the thin metal parts and is advantageous in terms of reliability and efficiency. There are two major operations in each assembly cycle: pickup and insertion. Under the configuration shown in Fig. 27.7a, all the operations are performed by the same workcell. Complex packaging operations must often be decomposed and performed by multiple workcells [27.76].

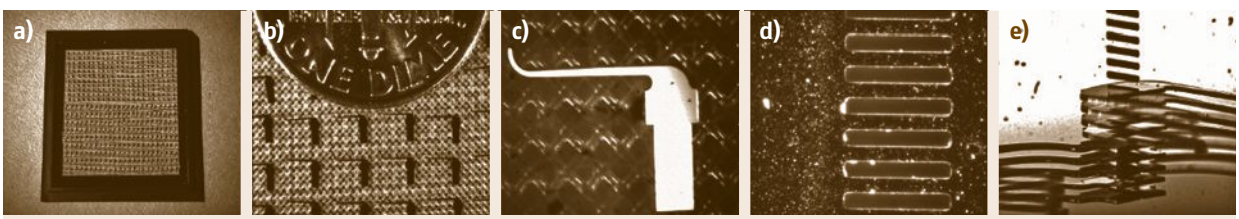


Fig.27.6a–e A 3-D microassembly example **(a)** Parts on release tray **(b)** parts versus a US dime **(c)** part shape **(d)** DRIE holes **(e)** assembly results

Avidbots Ex1016
Page 720 of 2154

General System Architecture

An automated microassembly system typically consists of the following functional units.

Large Workspace Positioning Unit. A large workspace and long-range positioning motion are required in most microassembly operations. The large workspace is necessary to accommodate different functional units, part feeders, and various tools.

In general, off-the-shelf motion control systems developed for automated IC packaging equipment can be adopted directly. For the task described in Sect. 27.6.1, the DRIE-etched holes are distributed on wafers up to 8 inches in diameter. This requires the assembly system to have a commensurate working space and high positioning speed. The coarse positioning unit has four DOFs (Fig. 27.7a). Planar motion in the horizontal direction is provided by an open-frame high-precision *XY* table with a travel of 32 cm (12 inch) and a repeatability of 1 μm in both directions. Position feedback with a resolution of 0.1 μm is provided by two linear encoders. A dual-loop PID (proportional–integral–derivative) plus feedforward control scheme is used for each axis. The internal speed loop is closed on the rotary encoder on the motor. The external position loop is closed on the linear encoder. Each wafer is placed on the vertical wafer mount that provides both linear and rotational control (Fig. 27.7b). Vertical motions of the wafer mount are provided by a linear slide with a travel of 20 cm (8 inch) and a repeatability of 5 μm. It is also controlled using a PID plus feedforward algorithm. Both the *XY* table and the vertical linear slides are actuated using alternating-current (AC) servo motors. The rotation of the wafer mount is actuated by an Oriental PK545AUA microstep motor with a maximum resolution of 0.0028° step. All low-level controllers are commanded and coordinated by a host computer [27.61].

For applications requiring repeatability of one micron or greater, it is convenient to use conventional positioning tables to implement coarse range motions. These tables typically use leadscrew or ballscrew drives and ball or roller bearings. For applications requiring submicron or nanometer repeatability, a few solutions are also commercially available. For example, piezoactuators are often used for nanometer repeatability motion. The disadvantage of piezoactuators is that their travel range is small, typically on the order of 100 microns. As another example, a series of positioning stages with submicron repeatability based on direct-drive linear actuators and air bearings is available from Aerotech. It is also possible to use parallel structure mechanisms such as a Stewart platform [27.77]. In general, the development of IC manufacturing towards the deep submicron level provides a major driving force behind the development of these motion control techniques.

Micromanipulator Unit. Fine pose (position and orientation) control is required in operations such as 3-D precision alignment and assembly. For example, the six DOFs required by the task introduced in Sect. 27.6.1 are implemented on separate structures. The three Cartesian DOF are provided by an adapted Sutter MP285 micromanipulator, which also provides yaw motions with its rotational DOF (Fig. 27.7c). Roll motions are implemented on the wafer mount (Fig. 27.7b). The pitch movement of the metal part after pickup is not motorized and is implemented through manual adjustment and calibration before assembly. More discussion of micromanipulator configuration can be found in [27.61].

Two principles need to be considered in implementing motion control for automated 3-D microassembly. The first is the partition of large-workspace coarse positioning unit and the fine positioning micromanipulator unit. The second is the decomposition and distributed implementation of multiple DOFs. In practice, the actual implementations of these principles are highly application specific. For certain applications, if a large-range positioning unit is sufficient to satisfy assembly



Fig.27.7a–c An experimental microassembly workcell **(a)** System overview **(b)** assembly scene **(c)** micromanipulator unit

tolerance requirements, the separate implementation of a micromanipulator may even be unnecessary.

In general, the separation of a micromanipulator unit will facilitate the implementation of high-bandwidth motion control. However, this separation normally brings redundancy to the entire motion control system. The functioning of the high-precision micromanipulator must also rely on closed-loop feedback control, especially microscopic vision feedback.

Automated Microgripper Unit. The function of a microgripper is to provide geometrical and physical constraints (grasping) in pick-and-place and assembly operations. The reliability and efficiency of the microgripper is critical to the performance of the entire automated microassembly system. Several factors must be considered in micromanipulator design. First of all, as the micromanipulator end-effector, the microgripper must be constantly monitored under a microscope. Therefore, it must be small in size and suitable in shape to remain in the microscope's field of view and to minimize occlusion. Secondly, it is important to consider the different governing physics and to explore the use of various gripping forces [27.77]. Thirdly, microgrippers are used in both part pick-and-place and assembly. Since assembly operations typically require more constraints, microgrippers designed for pick-and-place operations may not necessarily be suitable for assembly operations. In fact, the microassembly task described in Sect. 27.6.2 is performed using a combined microgripper [27.61]. Microgripper development is often closely related to the development of fixtures that can also have microscale sizes.

Microscope Optics and Imaging Unit. The function of the microscope optics and imaging unit is to provide noncontact measurement of the geometry, motion, and spatial relations of assembly objects. Typical configurations of commercial device bonding systems use one or two vertical microscopes. An inverted microscope configuration is commonly used for backside alignment. On the other hand, 3-D microassembly may require two camera views in a stereo configuration. In the system shown in Fig. 27.7a, a total of four different views can be provided to its human operator: a global view of the entire assembly scene, a vertical microscopic view for part pickup, and two lateral microscopic views for the fine position and orientation adjustments during the final microassembly operations. Each view uses a CCD camera with a matching optical system. All images are captured using a Matrox Corona peripheral component interconnect (PCI) frame grabber.

Microscopic visual feedback is crucial for precise 3-D alignment. Provided that resolution requirements of the assembly task can be satisfied, microscope optics with larger working distances is desirable. For the assembly task addressed in Sect. 27.6.1, an Edmund Scientific VZM 450i zoom microscope with a 1× objective is used to provide the right view. Its working distance is approximately 90 mm, with a resolution of 7.5 μm. The vertical view is also provided by a VZM 450i microscope with a 0.5× objective to guide pickup operations. Its field of view can range from 2.8 × 2.8 mm to 17.6 × 17.6 mm. Its working distance is approximately 147 mm.

If visual servoing is required in automate assembly operations, a stereo configuration formed by adding another lateral view may be necessary. Higher resolution may be necessary in some microassembly tasks. In such cases, the same configuration can be used with higher-resolution microscope optics, for example, two Navitar TenX zoom microscopes with Mitutoyo ultralong-working-distance M Plan Apo 10× objectives have been used in this configuration by the authors. The resolution of each microscope is 1 μm, with a working distance of 33.5 mm. Consequently, the usable workspace of the micromanipulator is reduced [27.61].

The global view is implemented using a miniature Marshall V-1260 board camera to monitor the status of the entire assembly scene. It plays an important role in helping the operator to understand gross spatial relations and preventing operation errors.

From the perspective of robotic systems, the development of automation microassembly systems for complex 3-D microassembly operation will rely on the development of:

1. Compact, robust and high-speed micromanipulators with 5–6 DOFs.
2. Highly reliable and efficient microgrippers that are suitable for working under microscopes. Such grippers should have active force control or passive compliance to avoid damage to MEMS devices.
3. Three-dimensional microscopic computer vision techniques, three-dimensional microforce measurement and control techniques, and their integration.

27.6.3 Basic Microassembly Techniques

This section introduces a few supporting techniques important to automated microassembly systems, including machine vision techniques, microforce control techniques, and simulation verification of assembly strategy.

Machine Vision Techniques

Machine vision techniques are widely used in the semiconductor industry. The major difference between ma-

chine vision and general computer vision [27.78] is that, unlike natural objects and scenes, industrial objects and scenes can often be artificially designed and configured. This advantage often makes it possible to significantly reduce the complexity and enhance the robustness of vision techniques. The applications of machine vision techniques can generally be categorized into the following two classes based on the requirement of real-time processing.

Non-Time-Critical Vision Applications. These applications do not require visual feedback for high-bandwidth real-time control. Examples include object recognition and packaging quality inspection [27.79].

Time-Critical Vision Applications. These applications require real-time visual feedback. Examples include vision-guided pick-and-place, alignment, insertion, etc. [27.80]. An introduction to 3-D computer vision techniques can be found in [27.81]. A standard introduction to visual servoing techniques can be found in [27.82].

Several commercial software packages are available from suppliers such as Cognex, Coreco Imaging, and National Instruments.

Microforce Control Techniques

The theory of force control has been studied by the robotics community for more than 50 years [27.83, 84]. Several theoretical frameworks and many control algorithms have been proposed and experimentally verified. A variety of macroscale multiple-DOF force sensors have been developed.

Force control is also crucial to microassembly. For example, contact forces in device bonding must often be programmed and precisely controlled. In general, many of the macroscale force control techniques can be applied at micro/mesoscales. Force control for device bonding is essentially 1-D and involves force on the order of several newtons. On the other hand, in the manipulation of micro/mesoscale parts, the magnitude of interactive force typically ranges from millinewtons (10^{-3} N) to micronewtons (10^{-6} N). Forces of this magnitude are often referred to as *microforces*. A major technical challenge in implementing microforce control is the lack of multiple-DOF microforce sensors. A basic requirement for multiple-DOF microforce sensors is that they must be miniature in size. The manufacturing of these sensors normally requires the use of micromachining, including MEMS techniques. There are two major microforce sensing configurations.

Stand-Alone Force Sensor. The advantage of this configuration is that the sensor is general purpose and can be used with different microgrippers. Most macroscale multiple-DOF force sensors are of this type. However, this also requires the sensor to have sufficient structural stiffness to support the static load of microgripper, which is often significantly larger than the force resolution to be reached.

Embedded Force Sensor. Micro strain gages can be attached to microgrippers [27.85]. Force-sensitive materials can also be deposited on microgrippers. This configuration avoids the issue of static load. However, such force sensing capabilities are dependent on the microgripper design, which often is not necessarily optimal for the measurement of multiple-DOF microforce/torque.

Complex and high-precision microassembly tasks also require the integration of microscopic machine vision with microforce control. Simple integration techniques use a gating/switching scheme [27.72, 86]. For more integrated approaches, visual impedance can be used [27.73].

Simulation Verification of Assembly Strategies

In many microassembly tasks, the distances between adjacent features are often on the meso/microscale. Therefore, selecting the correct assembly sequence is important for collision avoidance. In addition, due to the limited working distance of microscopes, microassembly operations must often be performed in a limited space using micromanipulators. Collision avoidance is critical to avoiding equipment or device damage. Potential collisions can be found and avoided by using offline simulation software. Many commercially available offline robot programming tools can provide this function.

Microassembly Tool

The end-effector of a micromanipulator in an automated microassembly system is often in the form of a microgripper, whose reliability and efficiency greatly influence the reliability and efficiency of the entire system. The microgripper must often be as small as possible. Its design must also minimize potential damage to fragile MEMS parts. This often requires passive compliance in structural design.

Microgrippers with integrated MEMS actuators can be fabricated monolithically and thus can be more compact in size. Several physical effects are commonly used in MEMS actuators, including electrostatic force and piezoelectric force [27.87], shape memory alloy (SMA) [27.88, 89], and thermal deformation [27.63, 90]. Currently, the major limitation is that it is difficult for MEMS actuators to generate sufficient travel, force, and power output. Alternatively, another solution

is to provide actuation externally [27.61, 91] The advantage is that sufficient travel and force and power output can be more easily obtained. The major disadvantage is that the microgripper is less compact in size. This could become a major obstacle to its applications.

27.7 Microrobotics

Today, more and more microrobotic devices are enabling new applications in various fields. Besides microassembly, microrobotics can play important roles in other industry fields for manipulation, characterization, inspection, and maintenance, and in biotechnology for, e.g., manipulating cells, a field referred to as biomicrorobotics.

27.7.1 Introduction

Microrobotics is a field that combines the established theory and techniques of robotics with the exciting new tools provided by MEMS technology in order to create intelligent machines that operate at micron scales. As stated by authors reviewing the microrobotics field [27.31, 92], many *micro* terms such as *micromechatronics*, *micromechanism*, *micromachines*, and *microrobots* are used synonymously to indicate a wide range of devices whose function is related to a *small* scale; however, *small* scale is a relative term so a clearer definition is needed.

The obvious difference between a macrorobot and a microrobot is the size of the robot. Thus, one definition of a microrobot is *a device having dimensions smaller than classical watch-making parts (i. e., μm to mm) and having the ability to move, apply forces and manipulate objects in a workspace with dimensions in the micrometer or submicrometer range* [27.93]. However, in many cases it is important that the robot can move over much larger distances. This task-specific definition is quite wide and includes several types of very small robots as well as stationary micromanipulation systems, which are a few decimeters in size but can carry out very precise manipulation (in the micron or even nanometer range) [27.92].

Besides classification by task or size, microrobots can also be classified by their mobility and functionality [27.31, 94]. Many robots usually consist of sensors and actuators, a control unit, and an energy source.

Depending on the arrangement of these components, one can classify microrobots according to the following criteria: locomotive and positioning possibility (yes or no), manipulation possibility (yes or no), control type (wireless or tethered), and autonomy. Figure 27.8 illustrates 15 different possible microrobot configurations by combining the four criteria [27.31, 94].

As depicted in Fig. 27.8 (taken from [27.24]), the classification is dependent on the following microrobot components: the control unit (CU), the power source (PS), the actuators necessary for moving the robot platform (i. e., the robot drive for locomotion and positioning; AP), and the actuators necessary for operation (i. e., manipulation using robot arms and hands; AM). Besides the different actuation functions, sensory functions are also needed, for example, tactile sensors for microgrippers or charge-coupled device (CCD) cameras for endoscopic applications (compare Fig. 27.8d and a).

The ultimate goal is to create a fully autonomous, wireless mobile microrobot equipped with suitable microtools according to Fig. 27.8o. Because this is a very difficult task, a good start is to investigate the possibility of making silicon microrobot platforms that are steered and powered through wires, like the one in Fig. 27.8c, and to study their locomotion capability.

The majority of MEMS-based microrobotic devices developed so far could be categorized as moveable links – microcatheters [27.95, 96], according to Fig. 27.8a, or microgrippers [27.25], such as those in Fig. 27.8d, or the microgrippers [27.97, 98] shown in Fig. 27.8e. Among the research publications covering locomotive microrobots, most publications have addressed microconveyance systems (Fig. 27.8b) [27.99–102]. Robots using external sources for locomotion could be used (compare Fig. 27.8b,f,j,n). According to *Fatikow* and *Rembold* [27.92], several researchers are working on methods to navigate micromechanisms through human blood vessels; however, these microrobots are difficult to control. Examples of partially autonomous systems (compare Fig. 27.8j) are the concept for so-called *smart pills*. Centimeter-sized pills for sensing temperature and/or pH inside the body have been presented [27.103, 104] as well as pills equipped with video cameras [27.105]. The pill is swallowed and transported to the part of the body where one wants to measure or record a video sequence. The information of the measured parameter or the signals from the camera is then transmitted (telemetrically) out of the body. More sophisticated approaches involving actuators for drug delivery of various kinds have also been proposed [27.92, 104]. The position of the pill inside the body is located by an X-ray monitor or ultrasound.

Avidbots Ex1016
Page 724 of 2154

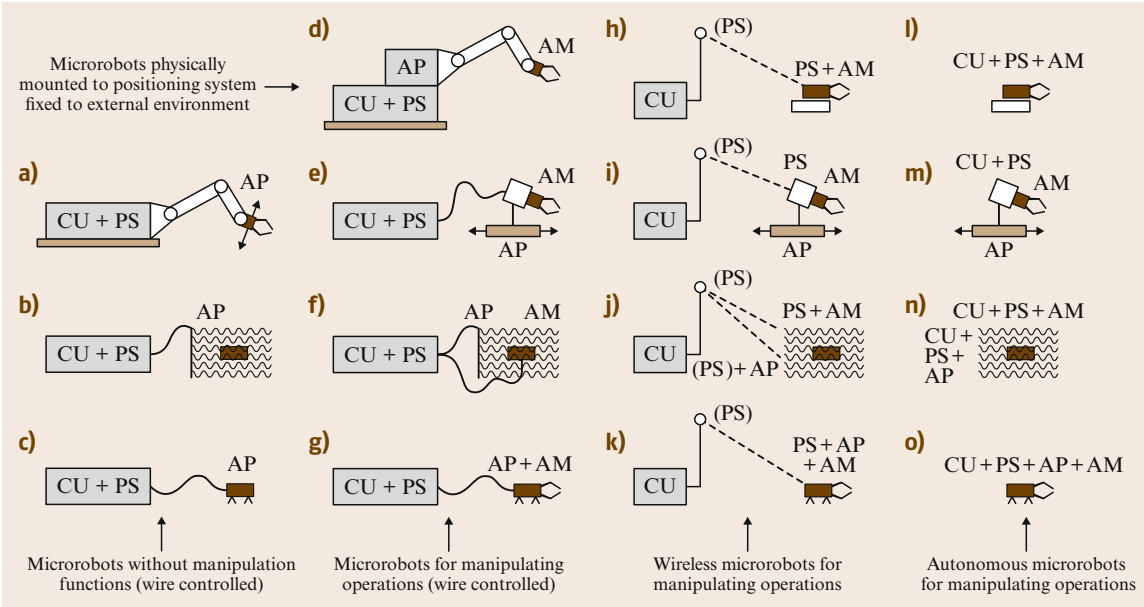


Fig.27.8a–o Classification of microrobots by functionality (modification of earlier presented classification schemes [27.12, 31, 92, 93]). (CU indicates the control unit; PS, the power source or power supply; AP, the actuators for positioning; AM, the actuators for manipulation): **(a–c)** Microrobots without manipulation functions (wire controlled), **(d–g)** microrobots for manipulation operations (wire controlled), **(h–k)** wireless microrobots for manipulation operations, **(l–o)** autonomous microrobots for manipulationg operations, **(d,h,l)** microrobots physically mounted to positioning system fixed to external environment

As soon as the pill reaches an infected area, a drug encapsulated in the pill can be released by the actuators onboard. External communication could be realized through radio signals.

Several important results have been presented regarding walking microrobots (Fig. 27.8c,g,k) fabricated by MEMS technologies and batch manufacturing. Different approaches for surface-micromachined robots [27.106, 107] and for a piezoelectric dry-reactive-ion-etched microrobot should be mentioned. A suitable low-power application-specific integrated circuit (ASIC) for robot control has been successfully tested and is planned to be integrated on a walking microrobot [27.108]. The large European Esprit project *MINIMAN* (1997) has the goal of developing moveable microrobotic platforms with integrated tools with six degrees of freedom for applications such as microassembly within an SEM and involves different MEMS research groups from several universities and companies across Europe. Further, miniature robot systems with MEMS/MST (microsystem technology) components have been developed [27.109].

Several research publications on *gnat* minirobots and actuator technologies for MEMS microrobots [27.3, 110] were reported by US researchers in the early 1990s, and several groups in Japan are also currently developing miniaturized robots based on MEMS devices [27.8]. In Japan, an extensive ten-year program on *micromachine technology*, supported by the Ministry of International Trade and Industry (MITI), started in 1991. One of the goals of this project is to create microsized and miniature robots for microfactory, medical technology, and maintenance applications. Several microrobotic devices, including locomotive robots and microconveyers, have been produced within this program. Miniature robot devices or vehicles [27.111] for locomotive tasks, containing several MEMS components, have been presented. Even though great efforts have been made on robot miniaturization using MEMS technologies, no experimental results on MEMS batch-fabricated microrobots suitable for autonomous walking (i. e., robust enough to be able to carry its own power source or to be powered by telemetric means) have been presented yet. The first batch-fabricated MEMS-based microrobot platform able to walk was presented in 1999 [27.112]. However, this robot was powered through wires and was not equipped with manipulation actuators. Besides walking microrobotic devices, several reports on flying [27.113–115] and swimming [27.116] robots have been published. Micromotors and gear boxes made using LIGA technology (a high-precision, lithograph-

ically defined plating technology) are used to build small flying microhelicopters, which are commercially available from the Institute of Microtechnology in Mainz, Germany, as rather expensive demonstration objects [27.114]. Besides the pure mechanical microrobots, hybrid systems consisting of electromechanical components and living organisms such as cockroaches have also been reported [27.117].

27.7.2 Bio-Microrobotics

Biomanipulation entails such operations as positioning, grasping, and injecting material into various locations in cells. Research topics in bio-microrobotics include the autonomous manipulation of single cells or molecules, the characterization of biomembrane mechanical properties using microrobotic systems with integrated vision and force sensing modules, and more. The objective is to obtain a fundamental understanding of single-cell biological systems and provide characterized mechanical models of biomembranes for deformable cell tracking during biomanipulation and cell injury studies.

Existing biomanipulation techniques can be classified into noncontact manipulation including laser trapping [27.118–121] and electrorotation [27.122–124], and contact manipulation, referred to as mechanical micromanipulation [27.125]. When laser trapping [27.118–121] is used for noncontact biomanipulation, a laser beam is focused through a large-numerical-aperture objective lens, converging to form an optical trap in which the lateral trapping force moves a cell in suspension toward the center of the beam. The longitudinal trapping force moves the cell in the direction of the focal point. The optical trap levitates the cell and holds it in position. Laser traps can work in a well-controlled manner. However, two features make laser trapping techniques undesirable for automated cell injection. The high dissipation of visible light in aqueous solutions requires the use of high-energy light close to the UV spectrum, raising the possibility of damage to the cell. Even though some researchers claim that such concerns could be overcome using wavelengths in the near-infrared (IR) spectrum [27.120], the question as to whether the incident laser beam might induce abnormalities in the cells' genetic material still exists. One alternative to using laser beams is the electrorotation technique. Electric-field-induced rotation of cells was demonstrated by *Mischel* et al. [27.126], *Arnold* and *Zimmermann* [27.127], and *Washizu* et al. [27.124]. This noncontact cell manipulation technique is based on controlling the phase shift and magnitude of electric fields. These fields, appropriately applied, produce a torque on the cell. Different system configurations have been established for cell manipulation based on this principle [27.122, 123], which can achieve high accuracy in cell positioning. However, it lacks a means to hold the cell in place for further manipulation, such as injection, since the magnitude of the electric fields has to be kept low to ensure the viability of cells. The limits of noncontact biomanipulation in the laser trapping and electrorotation techniques make mechanical micromanipulation desirable. The damage caused by laser beams in the laser trapping technique and the lack of a holding mechanism in the electrorotation technique can be overcome by mechanical micromanipulation.

To improve the low success rate of manual operation, and to eliminate contamination, an autonomous robotic system (shown in Fig. 27.9) has been developed to deposit DNA into one of the two nuclei of a mouse embryo without inducing cell lysis [27.11, 128]. The laboratory's experimental results show that the success rate for the autonomous embryo pronuclei DNA injection is dramatically improved over conventional manual

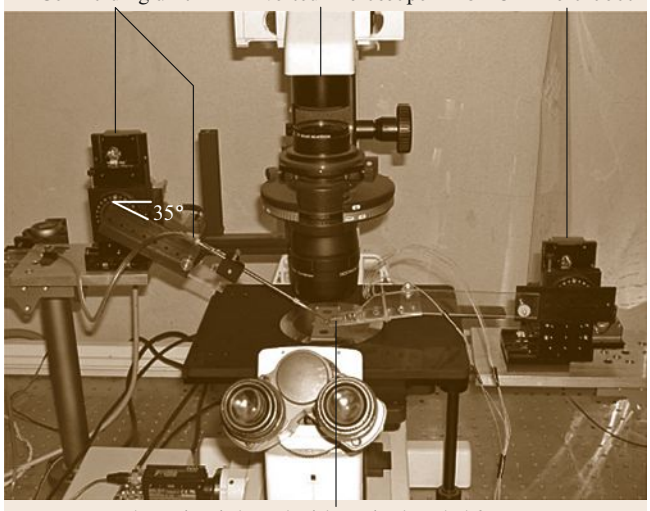


Fig. 27.9 Robotic biomanipulation system with vision and force feedback

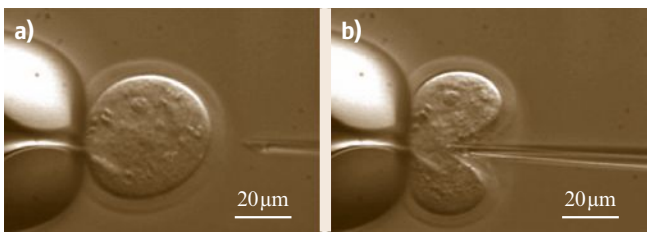


Fig. 27.10 Cell injection process. **(a)** Before and **(b)** during the injection of a mouse oocyte zona pellucida (ZP)

Avidbots Ex1016
Page 726 of 2154

injection methods. The autonomous robotic system features a hybrid controller that combines visual servoing and precision position control, pattern recognition for detecting nuclei, and a precise autofocusing scheme. Figure 27.10 illustrates the injection process.

To realize large-scale injection operations, a MEMS cell holder was fabricated using anodic wafer-bonding techniques. Arrays of holes are aligned on the cell holder, which are used to contain and fix individual cells for injection. When well calibrated, the system with the cell holder makes it possible to inject large numbers of cells using position control. The cell injection operation can be conducted in a move–inject–move manner.

A successful injection is determined greatly by injection speed and trajectory, and the forces applied to cells. To further improve the robotic system's performance, a multi-axial MEMS-based capacitive cellular force sensor is being designed and fabricated to provide realtime force feedback to the robotic system. The MEMS cellular force sensor also aids research in biomembrane mechanical property characterization.

MEMS-Based Multi-Axis Capacitive Cellular Force Sensor

The MEMS-based two-axis cellular force sensor [27.129] shown in Fig. 27.11 is capable of resolving normal forces applied to a cell as well as tangential forces generated by improperly aligned cell probes. A high-yield microfabrication process was developed to form the 3-D high-aspect-ratio structure using deep reactive ion etching (DRIE) on silicon-on-insulator (SOI) wafers. The constrained outer frame and the inner movable structure are connected by four curved springs. A load applied to the probe causes the inner structure to move, changing the gap between each pair of interdigitated comb capacitors. Consequently, the total capacitance change resolves the applied force. The interdigitated capacitors are orthogonally configured to make the force sensor capable of resolving forces in both the x- and y-directions. The cellular force sensors used in the experiments are capable of resolving forces up to 25 μ N with a resolution of 0.01 μ N.

Tip geometry affects the quantitative force measurement results. A standard injection pipette (Cook K-MPIP-1000-5) tip section with a tip diameter of 5 μ m is attached to the probe of the cellular force sensors.

The robotic system and high-sensitivity cellular force sensor are also applied to biomembrane mechanical property studies [27.130]. The goal is to obtain a general parameterized model describing cell membrane deformation behavior when an external load is applied. This parameterized model serves two chief purposes. First, in robotic biomanipulation, it allows online parameter recognition so that cell membrane deformation behavior can be predicted. Second, for a thermodynamic model of membrane damage in cell injury and recovery studies, it is important to appreciate the mechanical behavior of the membranes. This allows the interpretation of such reported phenomena as mechanical resistance to cellular volume reduction during dehydration, and its relationship to injury. The establishment of such a biomembrane model will greatly facilitate cell injury studies.

Experiments demonstrate that robotics and MEMS technology can play important roles in biological studies such as automating biomanipulation tasks. Aided by robotics, the integration of vision and force sensing modules, and MEMS design and fabrication techniques, investigations are being conducted in biomembrane mechanical property modeling, deformable cell tracking, and single-cell and biomolecule manipulation.

27.7.3 Bio-Mimetic/Untethered Microrobots

Similar to the larger scale, bio-mimicking is a significant approach for the design of micro- and nanorobots. However, at the micro-/nanoscale, the objectives we are learning from are not animals, fishes, birds, or insects anymore, rather we learn from such categories as bacteria, molecular motors, DNA molecules, and so on.

For instance, it is well known that a variety of microorganisms swim in liquid using flagella, a swimming strategy that is particularly well suited to their low-Reynolds-number regime [27.131]. Eukaryotic flagella

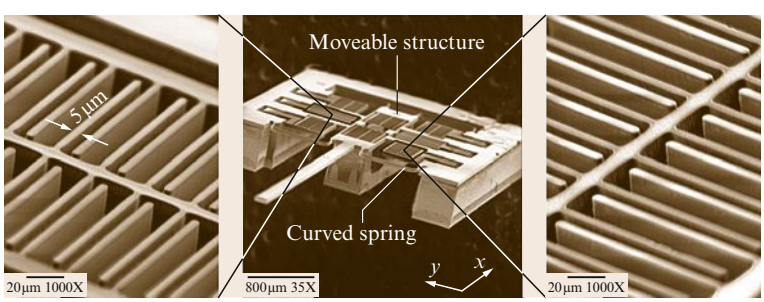


Fig. 27.11 A cellular force sensor with orthogonal comb drives detailed

Avidbots Ex1016
Page 727 of 2154

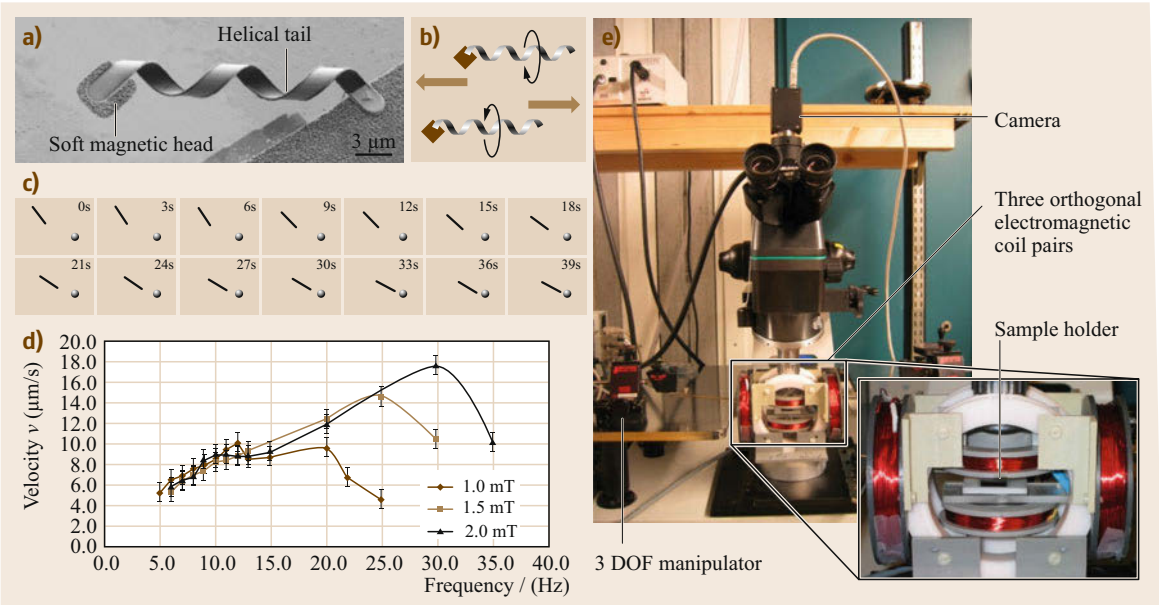


Fig.27.12a–e Artificial bacteria flagella. **(a)** SEM micrograph of an as-fabricated ABF with a diameter of 2.7 μm. **(b)** Schematic drawing of driving an ABF swims forward and backward as a rotary-to-linear motion converter. For forward and backward motion, the helical tail turns clockwise (CW) and counterclockwise (CCW), respectively (as seen by an observer in front of the ABF). **(c)** A series of frames taken from a video camera showing an ABF swim forward in water. The ABF has an 81 μm long InGaAs/GaAs/Cr helical tail, and a 4.5 $\mu m \times 4.5$ μm square head. B, f, θ, φ represent four parameters, i. e., magnetic field, field frequency, yaw and pitch, which can be controlled by the coil pairs. **(d)** The dependence of ABF velocity to frequency of external magnetic field generated by the coil pairs. **(e)** Experimental setup. All examples are from the authors' work

are active organelles that deform to create paddling motions, such as propagating waves or circular translating movements. Previous work experimentally demonstrated that this kind of motion can be approximated by a microscopic artificial swimmer consisting of a chain of DNA-bound magnetic beads connected to a red blood cell [27.132]. In contrast, bacterial (prokaryotic) flagella work differently by using a molecular motor to turn the base of a flagellum or bundle of flagella, which deform passively into a helical structure [27.133].

One example inspired by this is artificial bacterial flagella (ABFs) [27.134], which are proposed as microrobots capable of locomotion in liquid. An artificial bacterial flagellum consists of a helical *tail* resembling a natural flagellum, fabricated by the self-scrolling of helical nanobelts [27.135, 136], and a thin soft-magnetic *head*, consisting of a Cr/Ni/Au multilayer. Experimental investigation shows that an ABF can be propelled forward, backward and steered by a controlled external rotating magnetic field. ABFs have comparable swimming speed to bacteria propelled by flagella but under precise control.

An artificial bacterial flagellum (ABF) consists of two parts: a helical *tail* and a soft-magnetic metal *head* as shown in Fig. 27.12. Previous work has shown that the geometrical shape of the helical tail, i. e., chirality, helicity angle and diameter, can be precisely controlled. To control the locomotion of the ABF remotely, three orthogonal electromagnetic coil pairs were employed to generate a uniform rotating magnetic field [27.137].

In principle, for forward and backward motion, the ABF acts as a corkscrew to convert rotary motion to linear motion as shown in Fig. 27.12b. Figure 27.12c shows a left-handed ABF swim controlled by coil pairs. It can be seen in Fig. 27.12c that the ABF swim forward until the 10th second by rotating the magnet field clockwise (CW). In contrast to a left-handed ABF, when the same field is applied to a right-handed ABF, though it rotates in the same direction as left-handed one, the swimming direction will be opposite. The magnetic torque for rotation is generated by the thin head attempting to align with the applied field, and torque for steering is generated by the diagonal axis of the square head attempting to align with the applied field, since the diagonal axis is the easy magnetization axis of thin square plates in weak applied fields. Thus, to steer the ABFs in water, two more parameters were applied, i. e.,

yaw (θ) and pitch (φ), for the turning and tilting of the ABF.

The velocity of ABF is investigated as a function of magnetic field strength and rotation frequency in water (Fig. 27.12d. A 45 μm long ABF is used for the tests with three different field strengths: 1.0 mT, 1.5 mT and 2.0 mT. The field's rotational frequency is increased from 5 Hz to 35 Hz. The results show that at low frequency the swimmer's rotation is synchronized with the applied field, and ABF velocity increases linearly with frequency, as expected in the low-Reynolds-number regime [27.131]. After reaching a maximum value, the velocity reduces and becomes less deterministic with increasing field frequency, since the available magnetic torque is no longer sufficient to keep the swimmer synchronized with the applied field. The fluctuations in the curves are attributed to unmodeled boundary conditions, such as wall effects and intermolecular interactions of the ABF with the substrate. The maximum velocity we have achieved is 18 μm/s, which is comparable to bacteria, such as Escherichia coli [27.138], which swim by rotating their flagella with a frequency of about 100 Hz at room temperature. The results also indicate that by exerting a stronger magnetic torque on the ABF, higher driving frequencies can be achieved resulting in higher linear velocity. By extrapolating experimental results, if a frequency of 100 Hz can be achieved, the resulting swimming speed of ABFs may be beyond bacteria swimming speeds.

Self-propelled devices such as these are of interest in fundamental research and biomedical applications for mimicking and understanding natural organisms, for signaling inter- or intra-cellular information, for manipulating cellular or sub-cellular objects, and for targeted drug delivery. In contrast to bacteria that are propelled by nature flagella and move somewhat randomly in liquid, ABF motion can be precisely controlled by magnetic fields. ABFs can be used as a test bed for understanding the swimming behavior of bacteria and other swimming microorganisms, and can potentially be used for targeted drug delivery and as wireless manipulators for medical and biological applications.

27.8 Nanorobotics

Nanorobotics represents the next stage in miniaturization for maneuvering nanoscale objects. Nanorobotics is the study of robotics at the nanometer scale, and includes robots that are nanoscale in size, i. e., nanorobots, and large robots capable of manipulating objects that have dimensions in the nanoscale range with nanometer resolution, i. e., nanorobotic manipulators. Robotic manipulation at nanometer scales is a promising technology for structuring, characterizing, and assembling nanoscale building blocks into NEMS. Combined with recently developed nanofabrication processes, a hybrid approach is realized to build NEMS and other nanorobotic devices from individual carbon nanotubes and SiGe/Si nanocoils. Material science, biotechnology, electronics, and mechanical sensing and actuation will benefit from advances in nanorobotics.

27.8.1 Introduction

Nanomanipulation, or positional and/or force control at the nanometer scale, is a key enabling technology for nanotechnology by filling the gap between top-down and bottom-up strategies, and may lead to the appearance of replication-based molecular assemblers [27.18]. These types of assemblers have been proposed as general-purpose manufacturing devices for building a wide range of useful products as well as copies of themselves (self-replication).

Presently, nanomanipulation can be applied to the scientific exploration of mesoscopic physical phenomena, biology, and the construction of prototype nanodevices. It is a fundamental technology for property characterization of nanomaterials, nanostructures, and nanomechanisms, for the preparation of nanoscale building blocks, and for the assembly of nanodevices such as NEMS.

Nanomanipulation was enabled by the inventions of the STM [27.19], AFMs [27.44], and other types of scanning probe microscopes (SPMs). Besides these, optical tweezers (laser trapping) [27.139] and magnetic tweezers [27.140] are also potential nanomanipulators. Nanorobotic manipulators (NRMs) [27.141, 142] are characterized by the capability of 3-D positioning, orientation control, independently actuated multiple end-effectors, and independent real-time observation systems, and can be integrated with scanning probe microscopes. NRMs largely extend the complexity of nanomanipulation.

A concise comparison of STM, AFM, and NRM technology is shown in Fig. 27.13. With its incomparable imaging resolution, an STM can be applied to particles as small as atoms with atomic resolution. However, limited by its 2-D positioning and available strategies for manipulations, standard STMs are ill-suited for complex manipulation and cannot be used in 3-D space. An AFM is another important type of nanomanipulator. Manipulation with an AFM can

be done in either contact or dynamic mode. Generally, manipulation with an AFM involves moving an object by touching it with a tip. A typical manipulation starts by imaging a particle in noncontact mode, then removing the tip oscillation voltage and sweeping the tip across the particle in contact with the surface and with the feedback disabled. Mechanical pushing can exert larger forces on objects and, hence, can be applied for the manipulation of relatively larger objects. 1- to 3-D objects can be manipulated on a 2-D substrate. However, the manipulation of individual atoms with an AFM remains a challenge. By separating the imaging and manipulation functions, nanorobotic manipulators can have many more degrees of freedom including rotation for orientation control, and hence can be used for the manipulation of 0-D (symmetric spheres) to 3-D objects in 3-D free space. Limited by the lower resolution of electron microscopes, NRMs are difficult to use for the manipulation of atoms. However, their general robotic capabilities, including 3-D positioning, orientation control, independently actuated multiple endeffectors, separate real-time observation system, and integration with SPMs inside, make NRMs quite promising for complex nanomanipulation.

The first nanomanipulation experiment was performed by *Eigler* and *Schweizer* in 1990 [27.143]. They used an STM and materials at low temperatures (4 K) to position individual xenon atoms on a single-crystal nickel surface with atomic precision. The manipulation enabled them to fabricate rudimentary structures of their own design, atom by atom. The result is the famous set of images showing how 35 atoms were moved to form the three-letter logo *IBM*, demonstrating that matter could indeed be maneuvered atom by atom as *Feynman* suggested [27.1].

A nanomanipulation system generally includes nanomanipulators as the positioning device, microscopes as *eyes*, various end-effectors including probes and tweezers among others as its *fingers*, and various types of sensors (force, displacement, tactile, strain, etc.) to facilitate the manipulation and/or to determine the properties of the objects. Key technologies for nanomanipulation include observation, actuation, measurement, system design and fabrication, calibration and control, communication, and the human–machine interface.

Strategies for nanomanipulation are basically determined by the environment – air, liquid or vacuum – which is further decided by the properties and size of the objects and observation methods. Figure 27.14 depicts the microscopes, environments, and strategies of nanomanipulation. In order to observe manipulated objects, STMs can provide sub-Ångström imaging resolution, whereas AFMs can provide atomic resolutions. Both can obtain 3-D surface topology. Because AFMs can be used in an ambient environment, they provide a powerful tool for biomanipulation that may require a liquid environment. The resolution of SEM is limited to about 1 nm, whereas field-emission SEM (FESEM) can achieve higher resolutions. SEM/FESEM can be used for real-time 2-D observation for both the objects and end-effectors of manipulators, and large ultrahigh-vacuum (UHV) sample chambers provide enough space to contain an NRM with many degrees of freedom

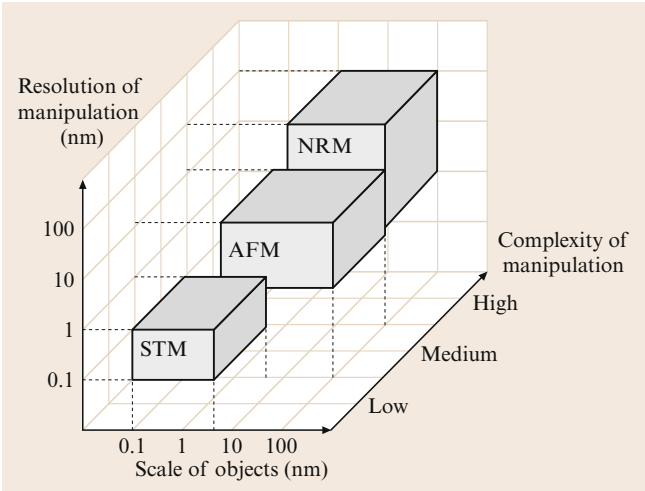


Fig. 27.13 Comparison of nanomanipulators

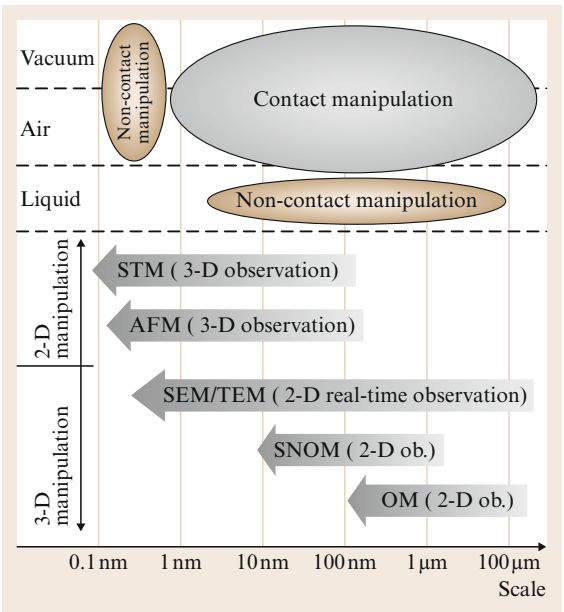


Fig. 27.14 Microscopes, environments, and strategies of nanomanipulation

(DOFs) for 3-D nanomanipulation. However, the 2-D nature of the observation makes positioning along the electron-beam direction difficult. High-resolution transmission electron microscopes (HRTEM) can provide atomic resolution. However, the narrow UHV specimen chamber makes it difficult to incorporate large manipulators. In principle, optical microscopes (OMs) cannot be used for nanoscale (smaller than the wavelength of visible lights) observation because of diffraction limits. Scanning near-field OMs (SNOMs) break this limitation and are promising as a real-time observation device for nanomanipulation, especially for ambient environments. SNOMs can be combined with AFMs, and potentially with NRMs for nanoscale biomanipulation.

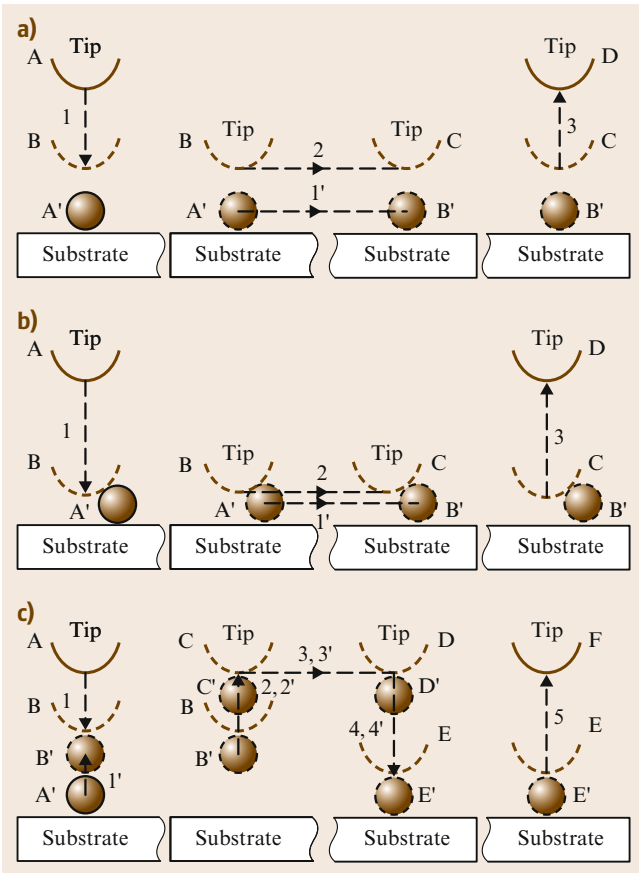


Fig.27.15a–c Basic strategies of nanomanipulation. In the figure, A, B, C,... represent the positions of end-effector (e.g., a tip); A′, B′, C′,... the positions of objects; 1, 2, 3,... the motions of end-effector; and 1′, 2′, 3′,... the motions of objects. Tweezers can be used in pick-and-place to facilitate the picking-up, but are generally not necessarily helpful for placing. **(a)** Lateral noncontact nanomanipulation (sliding). **(b)** Lateral contact nanomanipulation (pushing/pulling). **(c)** Vertical nanomanipulation (picking and placing)

Nanomanipulation processes can be broadly classified into three types:

1. Lateral noncontact
2. Lateral contact
3. Vertical manipulation.

Generally, lateral noncontact nanomanipulation is mainly applied for atoms and molecules in UHV with an STM or bio-object in liquid using optical or magnetic tweezers. Contact nanomanipulation can be used in almost any environment, generally with an AFM, but is difficult for atomic manipulation. Vertical manipulation can be performed by NRMs. Figure 27.15 shows the processes of the three basic strategies.

Motion of the lateral noncontact manipulation processes is shown in Fig. 27.15a. Applicable effects [27.144] able to cause the motion include long-range van der Waals (vdW) forces (attractive) generated by the proximity of the tip to the sample [27.145], electric-field-induced fields by the voltage bias between the tip and the sample [27.146, 147], tunneling current local heating or inelastic tunneling vibration [27.148, 149]. With these methods, some nanodevices and molecules have been assembled [27.150, 151]. Laser trapping (optical tweezers) and magnetic tweezers are possible for noncontact manipulation of nanoscale biosamples, e.g., DNA [27.152, 153].

Noncontact manipulation combined with STMs has revealed many possible strategies for manipulating atoms and molecules. However, for the manipulation of carbon nanotubes (CNTs) no examples have been demonstrated.

Pushing or pulling nanometer objects on a surface with an AFM is a typical manipulation using this method as shown in Fig. 27.15b. Early work showed the effectiveness of this method for the manipulation of nanoparticles [27.154–158]. This method has also been shown in nanoconstruction [27.159] and biomanipulation [27.160]. A virtual-reality interface facilitates such manipulation [27.161–163] and may create an opportunity for other types of manipulation. This technique has been used in the manipulation of nanotubes on a surface, and some examples will be introduced later in this chapter.

The pick-and-place task as shown in Fig. 27.15c is especially significant for 3-D nanomanipulation since its main purpose is to assemble prefabricated building blocks into devices. The main difficulty is in achieving sufficient control of the interaction between the tool and object and between the object and the substrate. Two strategies have been presented for micromanipulation [27.164] and have also proven to be effective for nanomanipulation [27.142]. One strategy is to apply

a dielectrophoretic force between a tool and an object as a controllable additional external force by applying a bias between the tool and the substrate on which the object is placed. Another strategy is to modify the van der Waals and other intermolecular and surface forces between the object and the substrate. For the former, an AFM cantilever is ideal as one electrode to generate a nonuniform electrical field between the cantilever and the substrate.

27.8.2 Nanorobotic Manipulation Systems

Nanorobotic manipulators are the core components of nanorobotic manipulation systems. The basic requirements for a nanorobotic manipulation system for 3-D manipulation include nanoscale positioning resolution, a relative large working space, enough DOFs including rotational ones for 3-D positioning and orientation control of the end-effectors, and usually multiple end effectors for complex operations.

A commercially available nanomanipulator (MM3A from Kleindiek) installed inside an SEM (Carl Zeiss DSM962) is shown in Fig. 27.16. The manipulator has three degrees of freedom, and nanometer- to subnanometer-scale resolution (Table 27.4). Calculations show that when moving/scanning in the A/B direction at joint q_1/q_2, the additional linear motion in C is very small. For example, when the arm length is 50 mm, the additional motion in the C direction is only 0.25 nm to 1 nm when moving 5–10 μm; in the A direction; these errors can be ignored or compensated with an additional motion of the prismatic joint p_3, which has a 0.25 nm resolution.

Figure 27.17a shows a nanorobotic manipulation system that has 16 DOFs in total and can be equipped with three or four AFM cantilevers as end-effectors for both manipulation and measurement. The positioning resolution is sub-nanometer and strokes are on the order

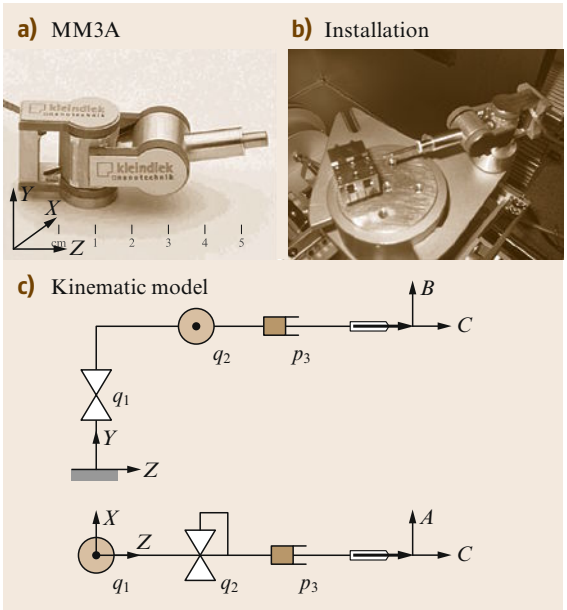


Fig. 27.16 **(a)** Nanomanipulator MM3A from Kleindiek **(b)** inside an SEM **(c)** kinematic model

Table 27.4 Specifications of MM3A

Item	Specification
Operating range q_1 and q_2	240°
Operating range Z	12 mm
Resolution A (horiz.)	10^{-7} rad (5 nm)
Resolution B (vert.)	10^{-7} rad (3.5 nm)
Resolution C (linear)	0.25 nm
Fine (scan) range A	20 μm
Fine (scan) range B	15 μm
Fine (scan) range C	1 μm
Speed A, B	10 mm/s
Speed C	2 mm/s

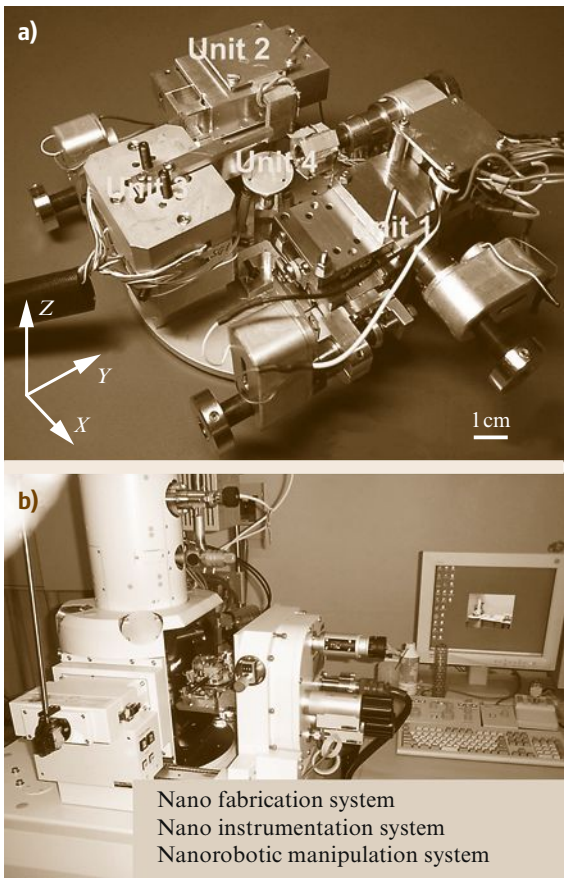


Fig.27.17a,b Nanorobotic system. **(a)** Nanorobotic manipulators. **(b)** System setup

Avidbots Ex1016
Page 732 of 2154

of centimeters. The manipulation system is not only for nanomanipulation, but also for nanoassembly, nanoinstrumentation, and nanofabrication. Four-probe semiconductor measurements are perhaps the most complex manipulation that this system can perform, because it is necessary to actuate four probes independently by using four manipulators. With the advancement of nanotechnology, one could shrink the size of nanomanipulators and insert more DOFs inside the limited vacuum chamber of a microscope, and perhaps the molecular version of manipulators such as that dreamed of by *Drexler* could be realized [27.18].

For the construction of multi-walled carbon nanotube (MWNT)-based nanostructures, manipulators position and orient nanotubes for the fabrication of nanotube probes and emitters, for performing nanosoldering with electron-beam-induced deposition (EBID) [27.165], for the property characterization of single nanotubes for selection purposes and for characterizing junctions to test connection strength.

A nanolaboratory is shown in Fig. 27.17b. The nanolaboratory integrates a nanorobotic manipulation system with a nanoanalytical system and a nanofabrication system, and can be applied for manipulating nanomaterials, fabricating nanoscale building blocks, assembling nanodevices, and for in situ analysis of the properties of such materials, building blocks, and devices. Nanorobotic manipulation within the nanolaboratory has opened a new path for constructing nanosystems in 3-D space, and will create opportunities for new nanoinstrumentation and nanofabrication processes.

27.8.3 Nanorobotic Manipulation and Assembly

Nanomanipulation is a promising strategy for nanoassembly. Key techniques for nanoassembly include the structuring and characterization of nanoscale building blocks, the positioning and orientation control of the building blocks with nanoscale resolution, and effective connection techniques. Nanorobotic manipulation, which is characterized by multiple DOFs with both position and orientation controls, independently actuated multiprobes, and a real-time observation system, have been shown effective for assembling nanotube-based devices in 3-D space.

The well-defined geometries, exceptional mechanical properties, and extraordinary electric characteristics, among other outstanding physical properties of CNTs [27.166] qualify them for many potential applications, especially in nanoelectronics [27.167], NEMS, and other nanodevices [27.168]. For NEMS, some of the most important characteristics of nanotubes include their nanometer diameter, large aspect ratio (10−1000), TPa-scale Young's modulus [27.169–171], excellent elasticity [27.141], ultralow interlayer friction, excellent capability for field emission, various electric conductivities [27.172], high thermal conductivity [27.173], high current-carrying capability with essentially no heating [27.174], sensitivity of conductance to various physical or chemical changes, and charge-induced bond-length change.

Helical 3-D nanostructures, or nanocoils, have been synthesized from various materials, including helical carbon nanotubes [27.175] and zinc oxide nanobelts [27.176]. A new method of creating structures with nanoscale dimensions has recently been presented [27.177] and can be fabricated in a controllable way [27.178, 179]. The structures are created through a top-down fabrication process in which a strained nanometer-thick heteroepitaxial bilayer curls up to form 3-D structures with nanoscale features. Helical geometries and tubes with diameters between 10 nm and 10 μm have been achieved. Because of their interesting morphology, mechanical, electrical, and electromagnetic properties, potential applications of these nanostructures in NEMS include nanosprings [27.180], electromechanical sensors [27.181], magnetic field detectors, chemical or biological sensors, generators of magnetic beams, inductors, actuators, and high-performance electromagnetic wave absorbers.

NEMS based on individual carbon nanotubes and nanocoils are of increasing interest, indicating that capabilities for incorporating these individual building blocks at specific locations on a device must be developed. Random spreading [27.182], direct growth [27.183], self-assembly [27.184], dielectrophoretic assembly [27.185, 186], and nanomanipulation [27.187] have been demonstrated for positioning as-grown nanotubes on electrodes for the construction of these devices. However, for nanotube-based structures, nanorobotic assembly is still the only technique capable of in situ structuring, characterization, and assembly. Because the as-fabricated nanocoils are not free-standing from their substrate, nanorobotic assembly is virtually the only way to incorporate them into devices at present.

Nanorobotic Assembly of Carbon Nanotubes

Nanotube manipulation in two dimensions on a surface was first performed with an AFM by contact pushing on a substrate. Figure 27.18 shows the typical methods for 2-D pushing. Although similar to that shown in Fig. 27.15b, the same manipulation caused various results because nanotubes cannot be regarded as a 0-D point. The first demonstration was given by *Wong* et al. for measuring the mechanical properties of

a nanotube [27.188]. They adopt the method shown in Fig. 27.18b, i. e., to bend a nanotube by pushing one end of it and fixing the other end. The same strategy was used for the investigation of the behavior of nanotubes under large strain [27.189]. *Postma* et al. applied the strategies shown in Fig. 27.18c,d to obtain a kinked junction and crossed nanotubes [27.190]. *Hertel* et al. combined this technique with an inverse process, namely straightening by pushing along a bent tube, and realized the translation of the tube to another location [27.191] and between two electrodes to measure the conductivity [27.192]. This technique was also used to place a tube on another tube to form a single-electron transistor (SET) with a cross-junction of nanotubes [27.193]. Pushing-induced breaking (Fig. 27.18d) has also been demonstrated for a nanotube [27.191]. The simple assembly of two bent tubes and a straight one formed a Greek letter θ. To investigate the dynamics of rolling at the atomic level, rolling and sliding of a nanotube (as shown in Fig. 27.18e,f) are performed on graphite surfaces using an AFM [27.194]. Besides pushing and pulling, another important process is indentation. By indenting a surface, mechanical property characterization [27.195] and data storage [27.196] can be realized.

Manipulation of CNTs in 3-D space is important for assembling CNTs into structures and devices. Basic techniques for the nanorobotic manipulation of carbon nanotubes are shown in Fig. 27.19 [27.197]. These serve as the basis for handling, structuring, characterizing, and assembling NEMS.

The basic procedure is to pick up a single tube from nanotube soot (Fig. 27.19a). This has been shown first by using dielectrophoresis [27.142] through nanorobotic manipulation (Fig. 27.19b). By applying a bias between a sharp tip and a plane substrate, a nonuniform electric field can be generated between the tip and the substrate with the strongest field near the tip. This field can cause a tube to orient along the field or further *jump* to the tip by electrophoresis or dielectrophoresis (determined by the conductivity of objective tubes). Removing the bias, the tube can be placed at other locations at will. This method can be used for free-standing tubes on nanotube soot or on a rough surface on which surface van der Waals forces are generally weak. A tube strongly rooted in CNT soot or lying on a flat surface cannot be picked up in this way. The interaction between a tube and the atomic flat surface of AFM cantilever tip has been shown to be strong enough for picking up a tube onto the tip [27.198] (Fig. 27.19c). By using EBID, it is possible to pick up and fix a nanotube onto a probe [27.199] (Fig. 27.19d). For handling a tube, weak connection between the tube and the probe is desired.

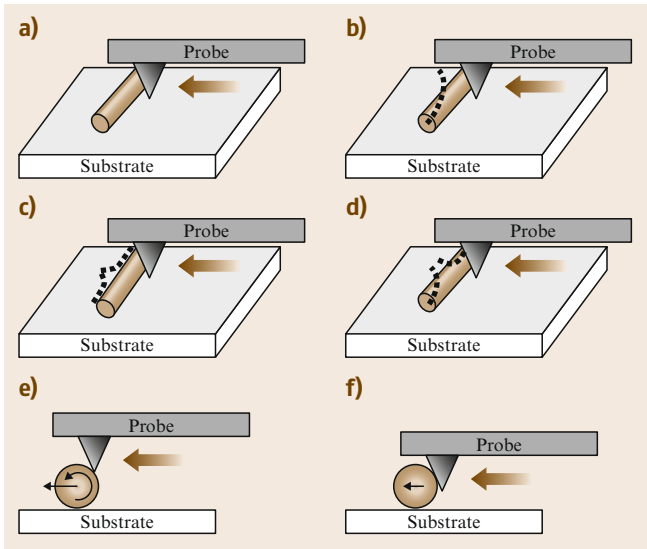


Fig.27.18a–f Two-dimensional manipulation of CNTs. Starting from the original state shown in **(a)**, pushing the tube at different site with different force may cause the tube to deform as in **(b)** and **(c)**, to break as in **(d)**, or to move as in **(e)** and **(f)**. **(a)** Original state. **(b)** Bending. **(c)** Kinking. **(d)** Breaking. **(e)** Rolling. **(f)** Sliding

Bending and buckling a CNT as shown in Fig. 27.19e,f are important for in situ property characterization of a nanotube [27.200], which is a simple way to obtain the Young's modulus of a nanotube without damaging the tube (if performed within its elastic range) and hence can be used for the selection of a tube with desired properties. By buckling an MWNT over its elastic limit, a kinked structure can be obtained [27.201]. To obtain any desired angle for a kinked junction it is possible to fix the shape of a buckled nanotube within its elastic limit by using EBID [27.202]. For a CNT, the maximum angular displacement will appear at the fixed left end under pure bending or at the middle point under pure buckling. A combination of these two kinds of loads will achieve a controllable position of the kinked point and a desired kink angle. If the deformation is within the elastic limit of the nanotube, it will recover as the load is released. To avoid this, EBID can be applied at the kinked point to fix the shape.

Stretching a nanotube between two probes or a probe and a substrate has generated several interesting results (Fig. 27.19g). The first demonstration of 3-D nanomanipulation of nanotubes took this as an example to show the breaking mechanism and to measure the tensile strength of CNTs [27.141]. By breaking an MWNT in a controlled manner, interesting nanodevices have been fabricated. This technique – destructive fabrication – has been presented to get sharpened and lay-

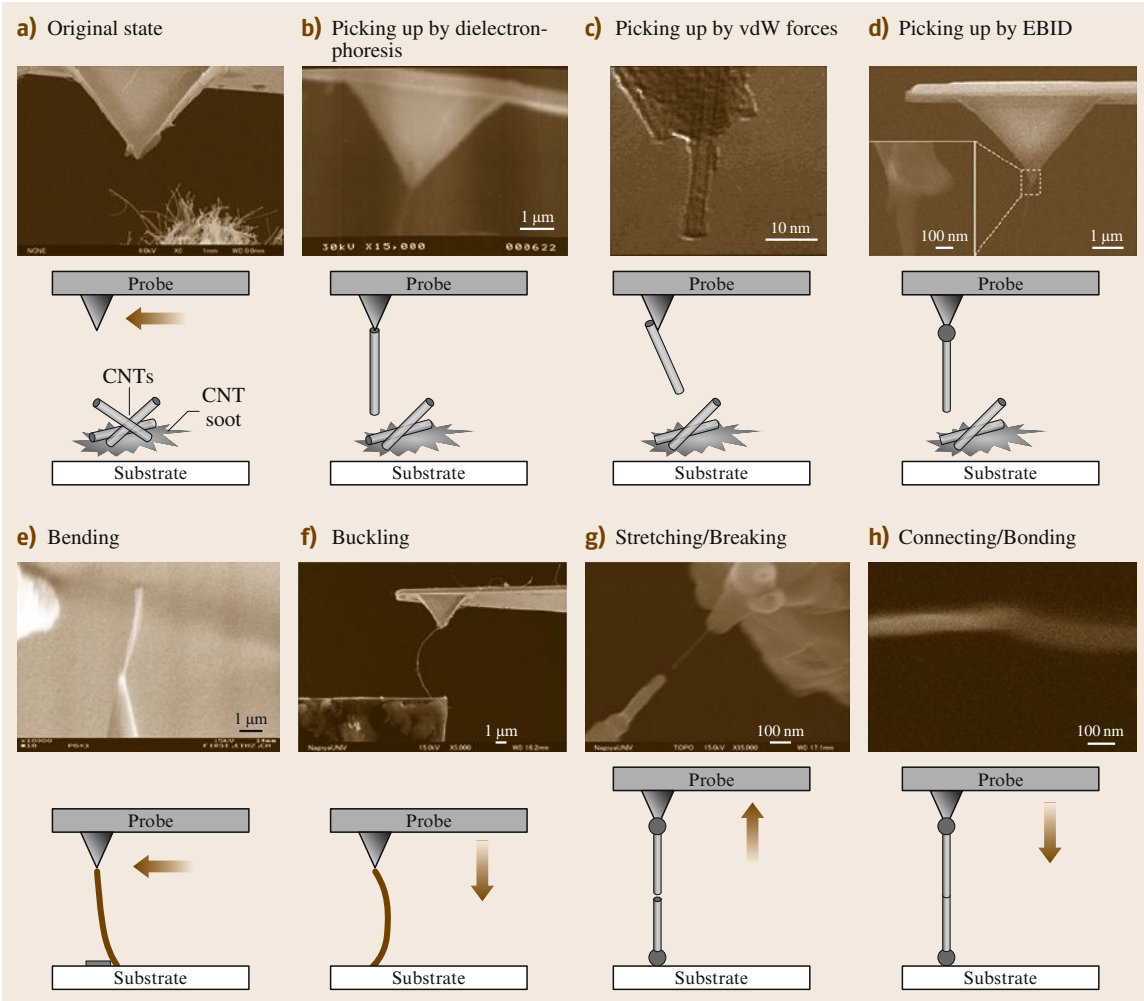


Fig.27.19a–h Nanorobotic manipulation of CNTs. The basic technique is to pick up an individual tube from CNT soot (as in **(a)**) or from an oriented array; **(b)** shows a free-standing nanotube picked up by dielectrophoresis generated by a nonuniform electric field between the probe and substrate, **(c)** (after [27.196]) and **(d)** show the same manipulation by contacting a tube with the probe surface or fixing (e.g., with EBID) a tube to the tip (inset shows the EBID deposit). Vertical manipulation of nanotubes includes bending **(e)**, buckling **(f)**, stretching/breaking **(g)**, and connecting/bonding **(h)**. All examples with the exception of **(i)** are from the authors' work

ered structures of nanotubes, and to improve the length control of a nanotube [27.201]. Typically, a layered and a sharpened structure can be obtained from this process, similar to that achieved from electric pulses [27.202]. Bearing motion has also been observed in an incompletely broken MWNT [27.201]. The interlayer friction has been shown to be very small [27.203, 204].

The reverse process, namely the connection of broken tubes (Fig. 27.19h), has been demonstrated recently, and the mechanism is revealed as rebonding of unclosed dangling bonds at the ends of broken tubes [27.205]. Based on this interesting phenomenon, mechanochemical nanorobotic assembly has been performed [27.206].

Assembly of nanotubes is a fundamental technology for enabling nanodevices. The most important tasks include the connection of nanotubes and placing of nanotubes onto electrodes. Pure nanotube circuits [27.206] created by interconnecting nanotubes of different diameters and chirality could lead to further size reductions in devices. Nanotube intermolecular and intramolecular junctions are basic elements for such systems [27.207]. Room-temperature (RT) single-electron transistors (SETs) [27.208] have been shown

with a short (≈ 20 nm) nanotube section that is created by inducing local barriers into the tube with an AFM, and Coulomb charging has been observed. With a cross-junction of two single-walled carbon nanotubes (SWNTs) (semiconducting/metallic), three- and four-terminal electronic devices have been made [27.209]. A suspended cross-junction can function as an electromechanical nonvolatile memory [27.210].

Although some kinds of junctions have been synthesized with chemical methods, there is no evidence yet showing that a self-assembly-based approach can provide more complex structures. SPMs were also used to fabricate junctions, but they are limited to a 2-D plane. We have presented 3-D nanorobotic manipulation-based nanoassembly, which is a promising strategy both for the fabrication of nanotube junctions and for the construction of more complex nanodevices with such junctions as well.

Nanotube junctions can be classified into different types by the kinds of components (SWNTs or MWNTs), geometric configurations (V kink, I, X cross, T, Y branch, and 3-D junctions) conductivity (metallic or semiconducting), and connection methods (intermolecular (connected with van der Waals force, EBID, etc.) or intramolecular (connected with chemical bonds) junctions). Here we show the fabrication of several kinds of MWNT junctions by emphasizing the connection methods. These methods will also be effective for SWNT junctions. Figure 27.20 shows CNT junctions constructed by connecting with van der Waals forces (a), joining by electron-beam-induced deposition (b), and bonding through mechanochemistry (c).

Figure 27.20a shows a T-junction connected with van der Waals forces, which is fabricated by positioning the tip of an MWNT onto another MWNT until they form a bond. The contact is checked by measuring the shear connection force.

EBID provides a soldering method to obtain stronger nanotube junctions than those connected through van der Waals forces. Hence, if the strength of nanostructures is emphasized, EBID can be applied. Figure 27.20b shows an MWNT junction connected through EBID, in which the upper MWNT is a single one with 20 nm in diameter and the lower one is a bundle of MWNTs with an extruded single CNT with Ø 30 nm. The development of conventional EBID has been limited by the expensive electron filament used and low productivity. We have presented a parallel EBID system by using CNTs as emitters because of their excellent field-emission properties [27.211]. The feasibility of parallel EBID is presented. It is a promising strategy for large-scale fabrications of nanotube junctions. As in its macroscale counterpart, welding, EBID works by adding materials to obtain stronger

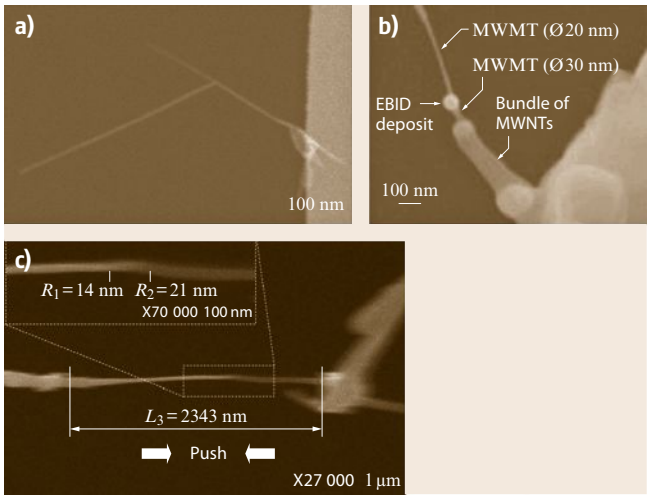


Fig.27.20a–c MWNT junctions. **(a)** MWNTs connected with van der Waals forces. **(b)** MWNTs joined with EBID. **(c)** MWNTs bonded with a mechanochemical reaction

connections, but in some cases, the added material might influence normal functions for nanosystems. So, EBID is mainly applied to nanostructures rather than nanomechanisms.

To construct stronger junctions without adding additional materials, mechanochemical nanorobotic assembly is an important strategy. Mechanochemical nanorobotic assembly is based on solid-phase chemical reactions, or mechanosynthesis, which is defined as chemical synthesis controlled by mechanical systems operating with atomic-scale precision, enabling direct positional selection of reaction sites [27.18]. By picking up atoms with dangling bonds rather than natural atoms only, it is easier to form primary bonds, which provides a simple but strong connection. Destructive fabrication provides a way to form dangling bonds at the ends of broken tubes. Some of the dangling bonds may close with neighboring atoms, but generally a few bonds will remain dangling. A nanotube with dangling bonds at its end will bind more easily to another to form intramolecular junctions. Figure 27.20c shows such a junction.

Three-dimensional nanorobotic manipulation has opened a new route for the structuring and assembly of nanotubes into nanodevices. However, at present nanomanipulation is still performed in a serial manner with master–slave control, which is not a large-scale production-oriented technique. Nevertheless, with advances in the exploration of mesoscopic physics, better control on the synthesis of nanotubes, more accurate actuators, and effective tools for manipulation, high-speed and automatic nanoassembly will be possible. Another approach might be parallel assembly by positioning building blocks with an array of probes [27.212] and

Avidbots Ex1016
Page 736 of 2154

joining them together simultaneously, e.g., with the parallel EBID [27.199] we presented. Further steps might progress towards exponential assembly [27.213], and in the far future to self-replicating assembly [27.18].

Nanorobotic Assembly of Nanocoils

The construction of nanocoil-based NEMS involves the assembly of as-grown or as-fabricated nanocoils, which is a significant challenge from a fabrication standpoint. Focusing on the unique aspects of manipulating nanocoils due to their helical geometry, high elasticity, single-end fixation, and strong adhesion of the coils to the substrate from wet etching, a series of new processes has been presented using a manipulator (MM3A, Kleindiek) installed in an SEM (Zeiss DSM962). As-fabricated SiGe/Si bilayer nanocoils (thickness 20 nm without Cr layer or 41 nm with Cr layer; diameter $D = 3.4\,\mu m$) are manipulated. Special tools have been fabricated including a nanohook prepared by controlled *tip-crashing* of a commercially available tungsten sharp probe (Picoprobe T-4-10–1 mm and T-4-10) onto a substrate, and a *sticky* probe prepared by tip dipping into a double-sided SEM silver conductive tape (Ted Pella, Inc.). As shown in Fig. 27.21, experiments demonstrate that nanocoils can be released from a chip by lateral pushing, picked up with a nanohook or a *sticky* probe, and placed between the probe/hook and another probe or an AFM cantilever (Nano-probe, NP-S). Axial pulling/pushing, radial compressing/releasing, and bending/buckling have also been demonstrated. These processes have shown the effectiveness of manipulation for the characterization of coil-shaped nanostructures and their assembly for NEMS, which have been otherwise unavailable.

Configurations of nanodevices based on individual nanocoils are shown in Fig. 27.22. Cantilevered nanocoils as shown in Fig. 27.22a can serve as nanosprings. Nanoelectromagnets, chemical sensors, and nanoinductors involve nanocoils bridged between two electrodes as shown in Fig. 27.22b. Electromechanical sensors can use a similar configuration but with one end connected to a moveable electrode as shown in Fig. 27.22c. Mechanical stiffness and electric conductivity are fundamental properties for these devices that must be further investigated.

As shown in Fig. 27.21h, axial pulling is used to measure the stiffness of a nanocoil. A series of SEM images are analyzed to extract the AFM tip displace-

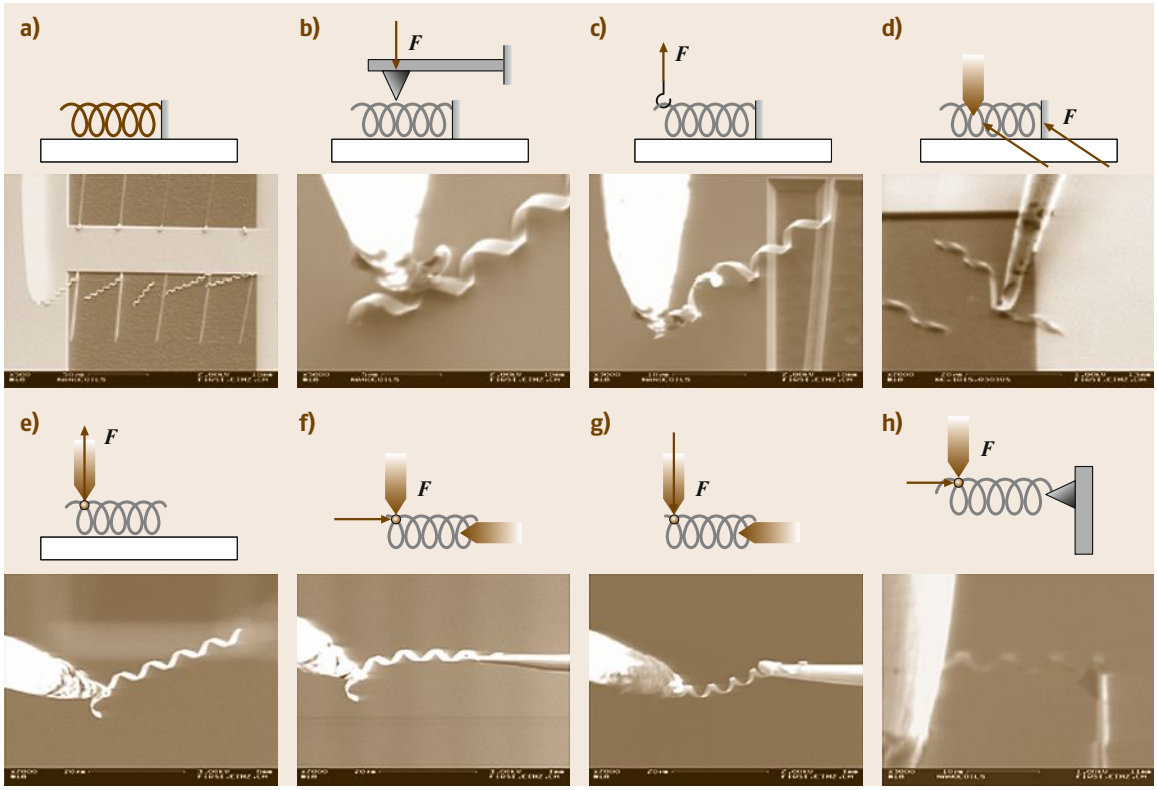


Fig.27.21a–h Nanorobotic manipulation of nanocoils **(a)** original state, **(b)** compressing/releasing, **(c)** hooking, **(d)** lateral pushing/breaking, **(e)** picking, **(f)** placing/inserting, **(g)** bending, and **(h)** pushing and pulling

ment and the nanospring deformation, i. e., the relative displacement of the probe from the AFM tip. From this displacement data and the known stiffness of the AFM cantilever, the tensile force acting on the nanospring versus the nanospring deformation was plotted. The deformation of the nanospring was measured relative to the first measurement point. This was necessary because the proper attachment of the nanospring to the AFM cantilever must be verified. Afterwards, it was not possible to return to the point of zero deformation. Instead, the experimental data as presented in Fig. 27.22d has been shifted such that with the calculated linear elastic spring stiffness the line begins at zero force and zero deformation. From Fig. 27.22d, the stiffness of the spring was estimated to be 0.0233 N/m. The linear elastic region of the nanospring extends to a deformation of 4.5 μm. An exponential approximation was fitted to the nonlinear region. When the applied force reached 0.176 μN, the attachment between the nanospring and the AFM cantilever broke. Finite-element simulation (ANSYS 9.0) was used to validate the experimental data [27.181]. Since the exact region of attachment cannot be identified from the SEM images, simulations were conducted for 4, 4.5, and 5 turns to estimate the possible range according to the apparent number of turns of the nanospring. The nanosprings in the simulations were fixed at one end and had an axial load of 0.106 μN applied at the other end. The simulation results for the spring with four turns yield a stiffness of 0.0302 N/m; for the nanospring with 5 turns it is 0.0191 N/m. The measured stiffness falls within this range with 22.0% above the minimum value and 22.8% below the maximum value, and very close to the stiffness of a 4.5-turn nanospring, which has a stiffness of 0.0230 N/m according to the simulation.

Figure 27.22e shows the results from electrical characterization experiments on a nanospring with 11 turns using the configuration shown in Fig. 27.21g. The *I–V* curve is nonlinear, which may be caused by the resistance change of the semiconductive bilayer due to ohmic heating. Another possible reason is the decrease in contact resistance caused by thermal stress. The maximum current was found to be 0.159 mA under an 8.8 V bias. Higher voltage causes the nanospring to *blow off*. From the fast scanning screen of the SEM, an extension of the nanospring on probes was observed around the peak current so that the current does not drop abruptly. At 9.4 V, the extended nanospring is broken down, causing an abrupt drop in the *I–V* curve.

From fabrication and characterization results, the helical nanostructures appear to be suitable to function as inductors. They would allow further miniaturization

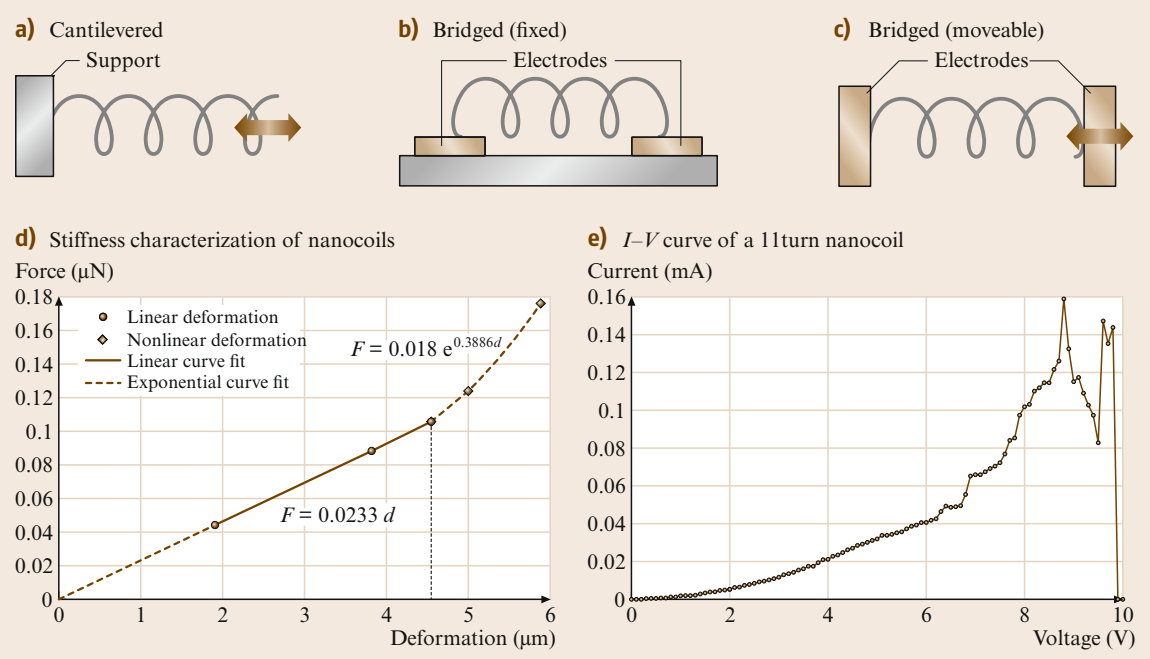


Fig.27.22a–e Nanocoil-based devices. Cantilevered nanocoils **(a)** can serve as nanosprings. Nanoelectromagnets, chemical sensors, and nanoinductors involve nanocoils bridged between two electrodes **(b)**. Electromechanical sensors can use a similar configuration but with one end connected to a moveable electrode **(c)**. Mechanical stiffness **(d)** and electric conductivity **(e)** are basic properties of interest for these devices

compared to state-of-the-art microinductors. For this purpose, higher doping of the bilayer and an additional metal layer would result in the required conductance. Conductance, inductance, and quality factor can be further improved if, after curling up, additional metal is electroplated onto the helical structures. Moreover, a semiconductive helical structure, when functionalized with binding molecules, can be used for chemical sensing under the same principle as demonstrated with other types of nanostructures. With bilayers in the range of a few monolayers, the resulting structures would exhibit very high surface-to-volume ratio with the whole surface exposed to an incoming analyst.

27.8.4 Nanorobotic Systems

Nanorobotic devices involve tools, sensors, and actuators at the nanometer scale. Shrinking device size makes it possible to manipulate nanosized objects with nanosized tools, measure mass in femtogram ranges, sense force at piconewton scales, and induce GHz motion, among other amazing advancements.

Top-down and bottom-up strategies for manufacturing such nanodevices have been independently investigated by a variety of researchers. Top-down strategies are based on nanofabrication and include technologies such as nanolithography, nanoimprinting, and chemical etching. Presently, these are 2-D fabrication processes with relatively low resolution. Bottom-up strategies are assembly-based techniques. At present, these strategies include such techniques as self-assembly, dip-pen lithography, and directed self-assembly. These techniques can generate regular nanopatterns at large scales. With the ability to position and orient nanoscale objects, nanorobotic manipulation is an enabling technology for structuring, characterizing and assembling many types of nanosystems [27.197]. By combining bottom-up and top-down processes, a hybrid nanorobotic approach based on nanorobotic manipulation provides a third way to fabricate NEMS by structuring as-grown nanomaterials or nanostructures. This new nanomanufacturing

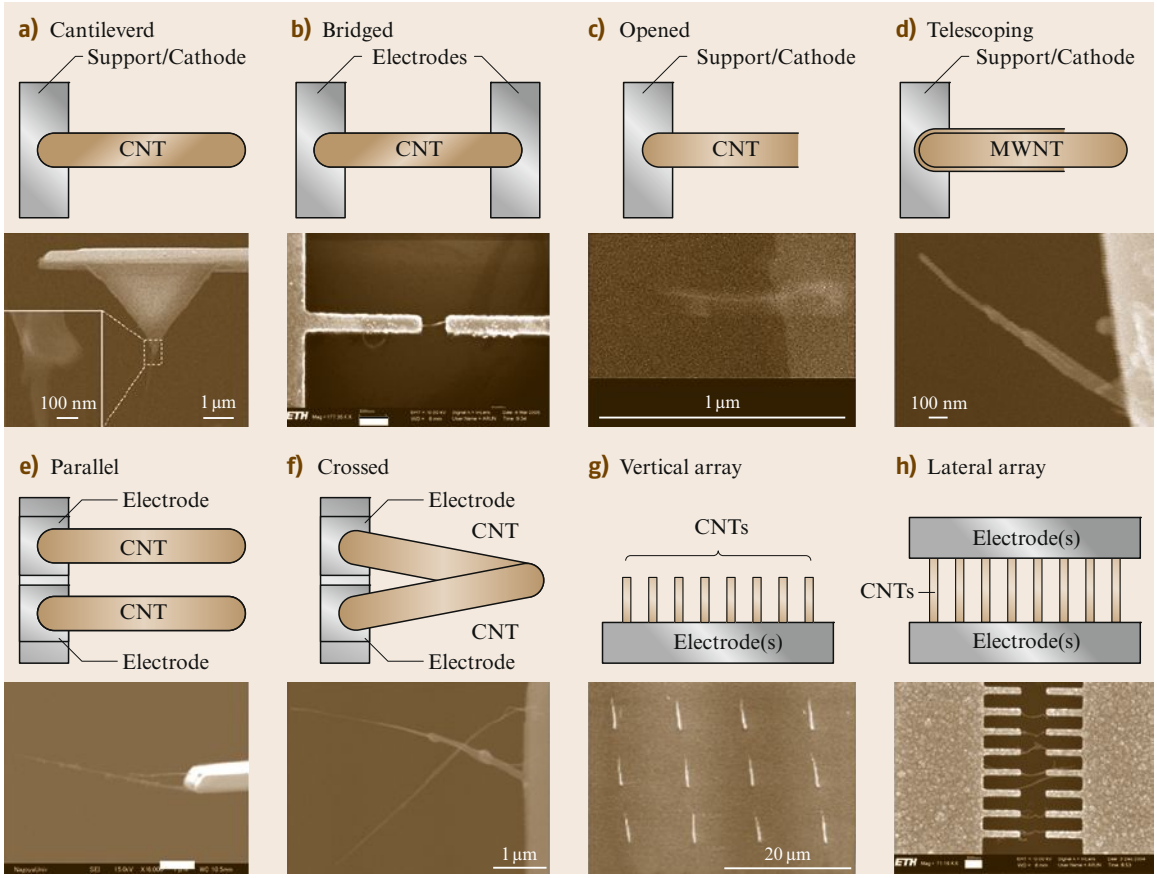


Fig.27.23a–h Configurations of individual nanotube-based NEMS. *Scale bars*: **(a)** 1 µm (*inset*: 100 nm), **(b)** 200 nm, **(c)** 1 µm, **(d)** 100 nm, **(e)** and **(f)** 1 µm, **(g)** 20 µm, and **(h)** 300 nm. All examples are from the authors' work

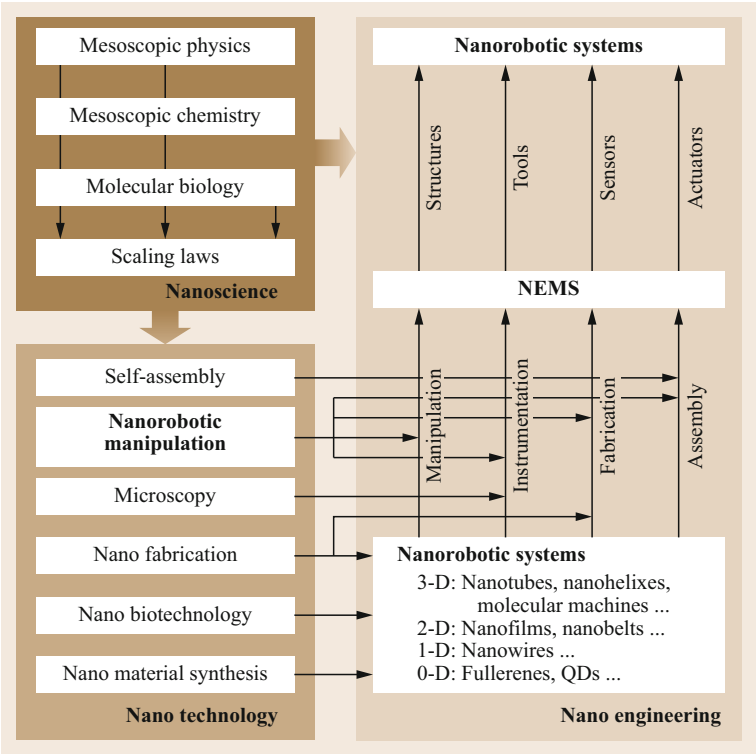


Fig. 27.24 A roadmap for nanorobotic systems

technique can be used to create complex 3-D nanodevices with such building blocks. Nanomaterial science, bio-nanotechnology, and nanoelectronics will also benefit from advances in nanorobotic assembly.

The configurations of nanotools, sensors, and actuators based on individual nanotubes that have been experimentally demonstrated are summarized as shown in Fig. 27.23.

For detecting deep and narrow features on a surface, cantilevered nanotubes (Fig. 27.23a, [27.199]) have been demonstrated as probe tips for an AFM [27.206], a STM, and other types of SPM. Nanotubes provide ultrasmall diameters, ultralarge aspect ratios, and excellent mechanical properties. Manual assembly [27.214], and direct growth [27.215] have proven effective for their construction. Cantilevered nanotubes have also been demonstrated as probes for the measurement of ultrasmall physical quantities, such as femtogram masses [27.170], piconewton-order force sensors, and mass flow sensors [27.197] on the basis of their static deflections or change of resonant frequencies detected within an electron microscope. Deflections cannot be measured from micrographs in real time, which limits the application of this kind of sensor. Interelectrode distance changes cause emission current variation of a nanotube emitter and may serve as a candidate to replace microscope images. Bridged individual nanotubes (Fig. 27.23b, [27.185]) have been the basis for electric characterization. Opened nanotubes (Fig. 27.23c, [27.216]) can serve as atomic or molecular containers, thermometers [27.217], or spot welders [27.218]. The electrostatic deflection of a nanotube has been used to construct a relay [27.219]. A new family of nanotube actuators can be constructed by taking advantage of the ultralow interlayer friction of a multiwalled nanotubes. Linear bearings based on telescoping nanotubes and a microactuator with a nanotube as a rotation bearing have been demonstrated [27.203, 220], and batch fabrication has been realized based on dielectrophoretically assembled arrays [27.221]. A preliminary experiment on a promising nanotube linear motor with field-emission current serving as position feedback has been shown with nanorobotic manipulation (Fig. 27.23d, [27.216]). Cantilevered dual nanotubes have been demonstrated as nanotweezers [27.222] and nanoscissors (Fig. 27.23e) [27.179] by manual and nanorobotic assembly, respectively. Based on electric resistance change under different temperatures, nanotube thermal probes (Fig. 27.23f) have been demonstrated for measuring the temperature at precise locations. These thermal probes are more advantageous than nanotube-based thermometers because the latter

require TEM imaging. The integration of the aforementioned devices can be realized using the configurations shown in Fig. 27.23g,h [27.183]. The arrays of individual nanotubes can also be used to fabricate nanosensors, such as position encoders [27.223].

Nanorobotic systems remains a rich research field with a large number of open problems. New materials such as nanowires, nanobelts, graphene, and polymer at the nanoscale will enable a new family of sensors and actuators for the detection and actuation of ultrasmall quantities or objects with ultrahigh precision and frequencies. Through random spreading, direct growth, optical tweezers, and nanorobotic manipulation, prototypes have been demonstrated. However, for integration into nanosystems, self-assembly processes will become increasingly important. Among them, we believe that dielectrophoretic nanoassembly will play a significant role for large-scale production of regular 2-D structures.

A roadmap for nanorobotic systems [27.224] is shown in Fig. 27.24. Knowledge from mesoscopic physics, mesoscopic/supramolecular chemistry, and molecular biology at the nanometer scale converges to form the field. Various disciplines contribute to nanorobotics, including nanomaterial synthesis, nanobiotechnology, and microscopy for imaging and characterization. Such topics as self-assembly, nanorobotic assembly, and hybrid nanomanufacturing approaches for assembling nano building blocks into structures, tools, sensors, and actuators are considered areas of nanorobotic study. A current focus of nanorobotics is on the fabrication of NEMS and other nanosystems, which may serve as components for future nanorobots. The main goals of nanorobotics are to provide effective tools for the experimental exploration of the nanoworld, and to push the boundaries of this exploration from a robotics research perspective.

27.9 Conclusions

Despite the claims of many *futurists*, such as Issac *Asimov*'s legendary submarine Proteus inside the human body [27.225] and *Robert A. Freitas's* nanomedical robots [27.226], the form that micro/nanorobots of the future will take and the tasks they will actually perform remain unclear. However, it is clear that technology is progressing towards the construction of intelligent sensors, actuators, and systems on small scales. These will serve as both the tools to be used for fabricating future micro/nanorobots as well as the components from which these robots may be developed. Shrinking device size to these dimensions presents many fascinating opportunities such as manipulating nano-objects with nanotools, measuring mass in femtogram ranges, sensing forces at piconewton scales, and inducing GHz motion, among other new possibilities waiting to be discovered. These capabilities will, of course, drive the tasks that future micro/nanorobots constructed by and with MEMS/NEMS will perform.

Video-References

VIDEO 11 Artificial bacterial flagella
available from http://handbookofrobotics.org/view-chapter/27/videodetails/11

VIDEO 12 The electromagnetic control of an untethered microrobot
available from http://handbookofrobotics.org/view-chapter/27/videodetails/12

VIDEO 13 A transversely magnetized rod-shaped microrobot
available from http://handbookofrobotics.org/view-chapter/27/videodetails/13

VIDEO 489 Attogram mass delivery from a carbon nanotube
available from http://handbookofrobotics.org/view-chapter/27/videodetails/489

VIDEO 490 Multi-beam bilateral teleoperation of holographic optical tweezers
available from http://handbookofrobotics.org/view-chapter/27/videodetails/490

VIDEO 491 High-speed magnetic microrobot actuation in a microfluidic chip by a fine V-groove surface
available from http://handbookofrobotics.org/view-chapter/27/videodetails/491

VIDEO 492 Linear-to-rotary motion converters for three-dimensional microscopy
available from http://handbookofrobotics.org/view-chapter/27/videodetails/492

References

27.1 R.P. Feynman: There's plenty of room at the bottom, Caltech Eng. Sci. **23**, 22–36 (1960)

27.2 R.S. Muller: Microdynamics, Sens. Actuators A **21**(1), 1–8 (1990)

27.3 A.M. Flynn, R.A. Brooks, W.M. Wells, D.S. Barrett: The world's largest one cubic inch robot, IEEE Micro Electro Mech. Syst. (MEMS) (1989) pp. 98–101

27.4 W. Trimmer, R. Jebens: Actuators for micro robots, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (1989) pp. 1547–1552

27.5 S. Fatikow, U. Rembold: An automated microrobot-based desktop station for micro assembly and handling of micro-objects, IEEE Conf. Emerg. Technol. Fact. Autom. (EFTA'96) (1996) pp. 586–592

27.6 B.J. Nelson, Y. Zhou, B. Vikramaditya: Sensor-based microassembly of hybrid MEMS devices, IEEE Control Syst. Mag. **18**, 35–45 (1998)

27.7 K. Suzumori, T. Miyagawa, M. Kimura, Y. Hasegawa: Micro inspection robot for 1-in pipes, IEEE/ASME Trans. Mechatron. **4**, 286–292 (1999)

27.8 M. Takeda: Applications of MEMS to industrial inspection, Proc. 14th IEEE Int. Conf. Micro Electro Mech. Syst. (MEMS) (2001) pp. 182–191

27.9 T. Frank: Two-Axis electrodynamic micropositioning devices, J. Micromech. Microeng. **8**, 114–118 (1989)

27.10 N. Kawahara, N. Kawahara, T. Suto, T. Hirano, Y. Ishikawa, T. Kitahara, N. Ooyama, T. Ataka: Microfactories: New applications of micromachine technology to the manufacture of small products, Res. J. Microsyst. Technol. **3**, 37–41 (1997)

27.11 Y. Sun, B.J. Nelson: Microrobotic cell injection, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (2001) pp. 620–625

27.12 P. Dario, M.C. Carrozza, L. Lencioni, B. Magnani, S. Dapos Attanasio: A micro robotic system for colonoscopy, Proc. Int. Conf. Robotics Autom. (ICRA) (1997) pp. 1567–1572

27.13 F. Tendick, S.S. Sastry, R.S. Fearing, M. Cohn: Application of micromechatronics in minimally invasive surgery, IEEE/ASME Trans. Mechatron. **3**, 34–42 (1998)

27.14 G. Iddan, G. Meron, A. Glukhovsky, P. Swain: Wireless capsule endoscopy, Nature **405**, 417 (2000)

27.15 K.B. Yesin, K. Vollmers, B.J. Nelson: Analysis and design of wireless magnetically guided microrobots in body fluids, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (2004) pp. 1333–1338

27.16 M.C. Roco, R.S. Williams, P. Alivisatos: *Nanotechnology Research Directions. Vision for Nanotechnology in the Next Decade* (Kluwer, Dordrecht 2000)

27.17 M.L. Downey, D.T. Moore, G.R. Bachula, D.M. Etter, E.F. Carey, L.A. Perine: *National Nanotechnology Initiative: Leading to the Next Industrial Revolution, A Report by the Interagency Working Group on Nanoscience, Engineering and Technology* (Committee on Technology, National Science and Technology Council, Washington 2000)

27.18 K. Drexler: *Nanosystems: Molecular Machinery, Manufacturing and Computation* (Wiley, New York 1992)

27.19 G. Binnig, H. Rohrer, C. Gerber, E. Weibel: Surface studies by scanning tunneling microscopy, Phys. Rev. Lett. **49**, 57–61 (1982)

27.20 W.F. Degrado: Design of peptides and proteins, Adv. Protein Chem. **39**, 51–124 (1998)

27.21 G.M. Whitesides, B. Grzybowski: Self-assembly at all scales, Science **295**, 2418–2421 (2002)

27.22 R. Fearing: Survey of sticking effects for micro-parts, Proc. IEEE/RSJ Int. Conf. Int. Robots Syst. (1995) pp. 212–217

27.23 E.L. Wolf: *Nanophysics and Nanotechnology* (Wiley-VCH, Weinheim 2004)

27.24 T. Ebefors, G. Stemme: Microrobotics. In: *The MEMS Handbook*, ed. by M. Gad-el-Hak (CRC, Boca Raton 2002)

27.25 C.-J. Kim, A.P. Pisano, R.S. Muller: Silicon-processed overhanging microgripper, IEEE/ASME J. Microelectromechanical Syst. **1**, 31–36 (1992)

27.26 C. Liu, T. Tsao, Y.-C. Tai, C.-M. Ho: Surface micromachined magnetic actuators, Proc. 7th IEEE Int. Conf. Micro Electro Mech. Syst. (MEMS) (1994) pp. 57–62

27.27 J. Judy, D.L. Polla, W.P. Robbins: A linear piezoelectric stepper motor with submicron displacement and centimeter travel, IEEE Trans. Ultrason. Ferroelectr. Freq. Control **37**, 428–437 (1990)

27.28 K. Nakamura, H. Ogura, S. Maeda, U. Sangawa, S. Aoki, T. Sato: Evaluation of the micro wobbler motor fabricated by concentric build-up process, Proc. 8th IEEE Int. Conf. Micro Electro Mech. Syst. (MEMS) (1995) pp. 374–379

27.29 A. Teshigahara, M. Watanabe, N. Kawahara, I. Ohtsuka, T. Hattori: Performance of a 7-mm microfabricated car, IEEE/ASME J. Microelectromechanical Syst. **4**, 76–80 (1995)

27.30 K.R. Udayakumar, S.F. Bart, A.M. Flynn, J. Chen, L.S. Tavrow, L.E. Cross, R.A. Brooks, D.J. Ehrlich: Ferroelectric thin film ultrasonic micromotors, Proc. 4th IEEE Int. Conf. Micro Electro Mech. Syst. (MEMS) (1991) pp. 109–113

27.31 P. Dario, R. Valleggi, M.C. Carrozza, M.C. Montesi, M. Cocco: Review – Microactuators for microrobots: A critical survey, J. Micromech. Microeng. **2**, 141–157 (1992)

27.32 I. Shimoyama: Scaling in microrobots, Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (1995) pp. 208–211

27.33 R.S. Fearing: Powering 3-dimensional microrobots: power density limitations, tutorial on Micro Mechatronics and Micro Robotics, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (1998)

27.34 R.G. Gilbertson, J.D. Busch: A survey of micro-actuator technologies for future spacecraft missions, J. Br. Interplanet. Soc. **49**, 129–138 (1996)

Avidbots Ex1016
Page 742 of 2154

27.35 M. Mehregany, P. Nagarkar, S.D. Senturia, J.H. Lang: Operation of microfabricated harmonic and ordinary side-drive motors, Proc. 3rd IEEE Int. Conf. Micro Electro Mech. Syst. (MEMS) (1990) pp. 1–8

27.36 Y.C. Tai, L.S. Fan, R.S. Mulle: IC-processed micromotors: design, technology, and testing, Proc. 2nd IEEE Int. Conf. Micro Electro Mech. Syst. (MEMS) (1989) pp. 1–6

27.37 T. Ohnstein, T. Fukiura, J. Ridley, U. Bonne: Micromachined silicon microvalve, Proc. 3rd IEEE Int. Conf. Micro Electro Mech. Syst. (MEMS) (1990) pp. 95–99

27.38 L.Y. Chen, S.L. Zhang, J.J. Yao, D.C. Thomas, N.C. MacDonald: Selective chemical vapor deposition of tungsten for microdynamic structures, Proc. 2nd IEEE Int. Conf. Micro Electro Mech. Syst. (MEMS) (1989) pp. 82–87

27.39 K. Yanagisawa, H. Kuwano, A. Tago: An electromagnetically driven microvalve, Proc. 7th Int. Conf. Solid-State Sens. Actuators (1993) pp. 102–105

27.40 M. Esashi, S. Shoji, A. Nakano: Normally close microvalve and micropump fabricated on a silicon wafer, Proc. 2nd IEEE Int. Conf. Micro Electro Mech. Syst. (MEMS) (1989) pp. 29–34

27.41 R. Petrucci, K. Simmons: An introduction to piezoelectric crystals, Sens. J. Appl.Sens. Technol. **11**(5), 26–31 (1994)

27.42 J. Goldstein, D. Newbury, D. Joy, C. Lyman, P. Echlin, E. Lifshin, L. Sawyer, J. Michael: *Scanning Electron Microscopy and X-ray Microanalysis* (Kluwer, New York 2003)

27.43 G. Binnig, H. Rohrer: In touch with atoms, Rev. Mod. Phys. **71**, S324–S330 (1999)

27.44 G. Binnig, C.F. Quate, C. Gerber: Atomic force microscope, Phys. Rev. Lett. **56**, 93–96 (1986)

27.45 M.J. Doktycz, C.J. Sullivan, P.R. Hoyt, D.A. Pelletier, S. Wu, D.P. Allison: AFM imaging of bacteria in liquid media immobilized on gelatin coated mica surfaces, Ultramicroscopy **97**, 209–216 (2003)

27.46 S.A. Campbell: *The Science and Engineering of Microelectronic Fabrication* (Oxford Univ. Press, New York 2001)

27.47 C.J. Jaeger: *Introduction to Microelectronic Fabrication* (Prentice Hall, Upper Saddle River 2002)

27.48 J.D. Plummer, M.D. Deal, P.B. Griffin: *Silicon VLSI Technology* (Prentice Hall, Upper Saddle River 2000)

27.49 M. Gad-el-Hak (Ed.): *The MEMS Handbook* (CRC, Boca Raton 2002)

27.50 T.-R. Hsu: *MEMS and Microsystems Design and Manufacture* (McGraw-Hill, New York 2002)

27.51 G.T.A. Kovacs: *Micromachined Transducers Sourcebook* (McGraw-Hill, New York 1998)

27.52 G.T.A. Kovacs, N.I. Maluf, K.A. Petersen: Bulk micromachining of silicon, Proc. IEEE Int. Conf. Robotics Autom. (1998) pp. 1536–1551

27.53 P. Rai-Choudhury (Ed.): *Handbook of Microlithography, Micromachining and Microfabrication* (SPIE, Bellingham 1997)

27.54 S.Y. Chou: Nano-imprint lithography and lithographically induced self-assembly, MRS Bulletin **26**, 512–517 (2001)

27.55 M.A. Herman: *Molecular Beam Epitaxy: Fundamentals and Current Status* (Springer, New York 1996)

27.56 J.S. Frood, G.J. Davis, W.T. Tsang: *Chemical Beam Epitaxy and Related Techniques* (Wiley, New York 1997)

27.57 S. Mahajan, K.S.S. Harsha: *Principles of Growth and Processing of Semiconductors* (McGraw-Hill, New York 1999)

27.58 C.A. Mirkin: Dip-pen nanolithography: automated fabrication of custom multicomponent, sub-100 nanometer surface architectures, MRS Bulletin **26**, 535–538 (2001)

27.59 C.A. Harper: *Electronic Packaging and Interconnection Handbook* (McGraw-Hill, New York 2000)

27.60 K.F. Bohringer, R.S. Fearing, K.Y. Goldberg: Microassembly. In: *Handbook of Industrial Robotics*, ed. by S. Nof (Wiley, New York 1999) pp. 1045–1066

27.61 G. Yang, J.A. Gaines, B.J. Nelson: A supervisory wafer-level 3D microassembly system for hybrid MEMS fabrication, J. Intell. Robotics Syst. **37**, 43–68 (2003)

27.62 P. Dario, M. Carrozza, N. Croce, M. Montesi, M. Cocco: Non-traditional technologies for microfabrication, J. Micromech. Microeng. **5**, 64–71 (1995)

27.63 W. Benecke: Silicon microactuators: activation mechanisms and scaling problems, Proc. IEEE Int. Conf. Solid-State Sens. Actuators (1991) pp. 46–50

27.64 A. Menciassi, A. Eisinberg, M. Mazzoni, P. Dario: A sensorized electro discharge machined superelastic alloy microgripper for micromanipulation: simulation and characterization, Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS) (2002) pp. 1591–1595

27.65 T.R. Hsu: Packaging design of microsystems and meso-scale devices, IEEE Trans. Adv. Packag. **23**, 596–601 (2000)

27.66 L. Lin: MEMS post-packaging by localized heating and bonding, IEEE Trans. Adv. Packag. **23**, 608–616 (2000)

27.67 A. Tixier, Y. Mita, S. Oshima, J.P. Gouy, H. Fujita: 3-D microsystem packaging for interconnecting electrical, optical and mechanical microdevices to the external world, Proc. 13th IEEE Int. Conf. Micro Electro Mech. Syst. (MEMS) (2000) pp. 698–703

27.68 M.J. Madou: *Fundamentals of Microfabrication* (CRC, Boca Raton 2002)

27.69 I. Shimoyama, O. Kano, H. Miura: 3D microstructures folded by Lorentz force, Proc. 11th IEEE Int. Conf. Micro Electro Mech. Syst. (MEMS) (1998) pp. 24–28

27.70 K.F. Bohringer, B.R. Donald, L. Kavraki, F.L. Lamiraux: Part orientation with one or two stable equilibria using programmable vector fields, IEEE Trans. Robot. Autom. **16**, 157–170 (2000)

27.71 V. Kaajakari, A. Lal: An electrostatic batch assembly of surface MEMS using ultrasonic triboelectric-

Avidbots Ex1016
Page 743 of 2154

ity, Proc. 14th IEEE Int. Conf. Micro Electro Mech. Syst. (MEMS) (2001) pp. 10–13
27.72 G. Yang, B.J. Nelson: Micromanipulation contact transition control by selective focusing and microforce control, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (2003) pp. 3200–3206
27.73 G. Morel, E. Malis, S. Boudet: Impedance based combination of visual and force control, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (1998) pp. 1743–1748
27.74 F. Arai, D. Andou, T. Fukuda: Adhesion forces reduction for micro manipulation based on micro physics, Proc. 9th IEEE Int. Conf. Micro Electro Mech. Syst. (MEMS) (1996) pp. 354–359
27.75 Y. Zhou, B.J. Nelson: The effect of material properties and gripping force on micrograsping, Proc. IEEE Int. Conf. Robotics Autom (ICRA) (2000) pp. 1115–1120
27.76 K. Kurata: Mass production techniques for optical modules, Proc. 48th IEEE Electronic Components and Technology Conf. (1998) pp. 572–580
27.77 V.T. Portman, B.-Z. Sandler, E. Zahavi: Rigid 6 ×6 parallel platform for precision 3-D micromanipulation: theory and design application, IEEE Trans. Robotics Autom. **16**, 629–643 (2000)
27.78 R.M. Haralick, L.G. Shapiro: *Computer and Robot Vision* (Addison-Wesley, Reading 1993)
27.79 A. Khotanzad, H. Banerjee, M.D. Srinath: A vision system for inspection of ball bonds and 2-D profile of bonding wires in integrated circuits, IEEE Trans. Semicond. Manuf. **7**, 413–422 (1994)
27.80 J.T. Feddema, R.W. Simon: CAD-driven microassembly and visual servoing, Proc. IEEE Int. Conf. Robotics Autom (ICRA) (1998), pp. 1212 -1219
27.81 E. Trucco, A. Verri: *Introductory Techniques for 3-D Computer Vision* (Prentice Hall, Upper Saddle River 1998)
27.82 S. Hutchinson, G.D. Hager, P.I. Corke: A tutorial on visual servo control, IEEE Trans. Robotics Autom. **12**, 651–670 (1996)
27.83 B. Siciliano, L. Villani: *Robot Force Control* (Kluwer, Dordrecht 2000)
27.84 T. Yoshikawa: Force control of robot manipulators, Proc. IEEE Int. Conf. Robotics Autom (ICRA) (2000) pp. 220–226
27.85 J.A. Thompson, R.S. Fearing: Automating microassembly with ortho-tweezers and force sensing, Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS) (2001) pp. 1327–1334
27.86 B.J. Nelson, P.K. Khosla: Force and vision resolvability for assimilating disparate sensory feedback, IEEE Trans. Robotics Autom. **12**, 714–731 (1996)
27.87 Y. Haddab, N. Chaillet, A. Bourjault: A microgripper using smart piezoelectric actuators, Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS) (2000) pp. 659–664
27.88 D. Popa, B.H. Kang, J. Sin, J. Zou: Reconfigurable micro-assembly system for photonics applications, Proc. IEEE Int. Conf. Robotics Autom (ICRA) (2002) pp. 1495–1500
27.89 A.P. Lee, D.R. Ciarlo, P.A. Krulevitch, S. Lehew, J. Trevin, M.A. Northrup: A practical microgripper by fine alignment, eutectic bonding and SMA actuation, Proc. IEEE Int. conf. Solid-State Sens. Actuators (1995) pp. 368–371
27.90 H. Seki: Modeling and impedance control of a piezoelectric bimorph microgripper, Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS) (1992) pp. 958–965
27.91 W. Nogimori, K. Irisa, M. Ando, Y. Naruse: A laser-powered micro-gripper, Proc. 10th IEEE Int. Conf. Micro Electro Mech. Syst. (MEMS) (1997) pp. 267–271
27.92 S. Fatikow, U. Rembold: *Microsystem Technology and Microrobotics* (Springer, Berlin, Heidelberg 1997)
27.93 T. Hayashi: Micro mechanism, J. Robotics Mechatr. **3**, 2–7 (1991)
27.94 S. Johansson: Micromanipulation for micro- and nanomanufacturing, INRIA/IEEE Symp. Emerging Technologies and Factory Automation (ETFA'95), Paris (1995) pp. 3–8
27.95 K.-T. Park, M. Esashi: A multilink active catheter with polyimide-based integrated CMOS interface circuits, J. Microelectromechanical Syst. **8**, 349–357 (1999)
27.96 Y. Haga, Y. Tanahashi, M. Esashi: Small diameter active catheter using shape memory alloy, Proc. IEEE 11th Int. Workshop on Micro Electro Mechanical Systems, Heidelberg (1998) pp. 419–424
27.97 E.W.H. Jager, O. Inganas, I. Lundstrom: Microrobots for micrometer-size objects in aqueous media: Potential tools for single cell manipulation, Science **288**, 2335–2338 (2000)
27.98 E.W.H. Jager, E. Smela, O. Inganas: Microfabricating conjugated polymer actuators, Science **290**, 1540–1545 (2000)
27.99 J.W. Suh, S.F. Glander, R.B. Darling, C.W. Storment, G.T.A. Kovacs: Organic thermal and electrostatic ciliary microactuator array for object manipulation, Sens. Actuators A **58**, 51–60 (1997)
27.100 E. Smela, M. Kallenbach, J. Holdenried: Electrochemically driven polypyrrole bilayers for moving and positioning bulk micromachined silicon plates, J. Microelectromechanical Syst. **8**, 373–383 (1999)
27.101 S. Konishi, H. Fujita: A conveyance system using air flow based on the concept of distributed micro motion systems, IEEE J. Microelectromechanical Syst. **3**, 54–58 (1994)
27.102 M. Ataka, A. Omodaka, N. Takeshima, H. Fujita: Fabrication and operation of polyimide bimorph actuators for a ciliary motion system, J. Microelectromechanical Syst. **2**, 146–150 (1993)
27.103 G.-X. Zhou: Swallowable or implantable body temperature telemeter-body temperature radio pill, Proc. IEEE 15th Annual Northeast Bioengineering Conf. (1989) pp. 165–166
27.104 A. Uchiyama: Endoradiosonde needs micro machine technology, Proc. IEEE 6th Int. Symp. Micro Mach. Hum. Sci. (MHS) (1995) pp. 31–37

Avidbots Ex1016
Page 744 of 2154

27.105 Y. Carts-Powell: *Tiny Camera in a Pill Extends Limits of Endoscopy*, OE-Rep. Aug., Vol. 200 (SPIE, Bellingham 2000)

27.106 R. Yeh, E.J.J. Kruglick, K.S.J. Pister: Surface-micromachined components for articulated microrobots, J. Microelectromechanical Syst. **5**, 10–17 (1996)

27.107 P.E. Kladitis, V.M. Bright, K.F. Harsh, Y.C. Lee: Prototype Microrobots for micro positioning in a manufacturing process and micro unmanned vehicles, Proc. IEEE 12th Int. Conf. Micro Electro Mech. Syst. (MEMS), Orlando (1999) pp. 570–575

27.108 D. Ruffieux, N.F. Rooij: A 3-DoF bimorph actuator array capable of locomotion, Proc. 13th Eur. Conf. Solid-State Transducers (Eurosensors), Hague (1999) pp. 725–728

27.109 J.-M. Breguet, P. Renaud: A 4 degrees-of-freedoms microrobot with nanometer resolution, Robotics **14**, 199–203 (1996)

27.110 A. Flynn, L.S. Tavrow, S.F. Bart, R.A. Brooks, D.J. Ehrlich, K.R. Udayakumar, L.E. Cross: Piezoelectric micromotors for microrobots, J. Microelectromechanical Syst. **1**, 44–51 (1992)

27.111 A. Teshigahara, M. Watanabe, N. Kawahara, Y. Ohtsuka, T. Hattori: Performance of a 7 mm microfabricated car, J. Microelectromechanical Syst. **4**, 76–80 (1995)

27.112 T. Ebefors, J. Mattson, E. Kalvesten, G. Stemme: A walking silicon micro-robot, 10th Int. Conf. Solid-State Sens. Actuators (Transducers), Sendai (1999) pp. 1202–1205

27.113 N. Miki, I. Shimoyama: Flight performance of micro-wings rotating in an alternating magnetic field, Proc. IEEE 12th Int. Conf. Micro Electro Mech. Syst. (MEMS), Orlando (1999) pp. 153–158

27.114 Mainz: Micro-motors: The World's Tiniest Helicopter, http://phys.org/news/2004-08-world-lightest-micro-flying-robot-built.html

27.115 K.I. Arai, W. Sugawara, T. Honda: Magnetic small flying machines, IEEE 8th Int. Conf. Solid-State Sens. Actuattors (1995) pp. 316–319

27.116 T. Fukuda, A. Kawamoto, F. Arai, H. Matsuura: Mechanism and swimming experiment of micro mobile robot in water, Proc. IEEE 7th Int. Workshop Micro Electro Mech. Syst. (MEMS), Oiso (1994) pp. 273–278

27.117 I. Shimoyama: Hybrid system of mechanical parts and living organisms for microrobots, Proc. IEEE 6th Int. Symp. Micro Mach. Hum. Sci. (MHS) (1995) p. 55

27.118 A. Ashkin: Acceleration and trapping of particles by radiation pressure, Phys. Rev. Lett. **24**, 156–159 (1970)

27.119 T.N. Bruican, M.J. Smyth, H.A. Crissman, G.C. Salzman, C.C. Stewart, J.C. Martin: Automated single-cell manipulation and sorting by light trapping, Appl. Opt. **26**, 5311–5316 (1987)

27.120 J. Conia, B.S. Edwards, S. Voelkel: The microrobotic laboratory: Optical trapping and scissing for the biologist, J. Clin. Lab. Anal. **11**, 28–38 (1997)

27.121 W.H. Wright, G.J. Sonek, Y. Tadir, M.W. Berns: Laser trapping in cell biology, IEEE J. Quant. Electron. **26**, 2148–2157 (1990)

27.122 F. Arai, K. Morishima, T. Kasugai, T. Fukuda: Bio-micromanipulation (new direction for operation improvement), Proc. IEEE/RSJ Int. Conf. Intell. Robotics Syst. (IROS) (1997) pp. 1300–1305

27.123 M. Nishioka, S. Katsura, K. Hirano, A. Mizuno: Evaluation of cell characteristics by step-wise orientational rotation using optoelectrostatic micromanipulation, IEEE Trans. Ind. Appl. **33**, 1381–1388 (1997)

27.124 M. Washizu, Y. Kurahashi, H. Iochi, O. Kurosawa, S. Aizawa, S. Kudo, Y. Magariyama, H. Hotani: Dielectrophoretic measurement of bacterial motor characteristics, IEEE Trans. Ind. Appl. **29**, 286–294 (1993)

27.125 Y. Kimura, R. Yanagimachi: Intracytoplasmic sperm injection in the mouse, Biol. Reprod. **52**, 709–720 (1995)

27.126 M. Mischel, A. Voss, H.A. Pohl: Cellular spin resonance in rotating electric fields, J. Biol. Phys. **10**, 223–226 (1982)

27.127 W.M. Arnold, U. Zimmermann: Electro-Rotation: Development of a technique for dielectric measurements on individual cells and particles, J. Electrost. **21**, 151–191 (1988)

27.128 Y. Sun, B.J. Nelson: Autonomous injection of biological cells using visual servoing, Int. Symp. Experim. Robotics (ISER) (2000) pp. 175–184

27.129 Y. Sun, B.J. Nelson, D.P. Potasek, E. Enikov: A bulk microfabricated multi-axis capacitive cellular force sensor using transverse comb drives, J. Micromech. Microeng. **12**, 832–840 (2002)

27.130 Y. Sun, K. Wan, K.P. Roberts, J.C. Bischof, B.J. Nelson: Mechanical property characterization of mouse zona pellucida, IEEE Trans. Nanobiosci. **2**, 279–286 (2003)

27.131 E.M. Purcell: Life at low Reynolds-number, Am. J. Phy. **45**, 3–11 (1977)

27.132 R. Dreyfus, J. Baudry, M.L. Roper, M. Fermigier, H.A. Stone, J. Bibette: Microscopic artificial swimmers, Nature **437**, 862–865 (2005)

27.133 H.C. Berg, R.A. Anderson: Bacteria swim by rotating their flagellar filaments, Nature **245**, 380–382 (1973)

27.134 L. Zhang, J.J. Abbott, L.X. Dong, B.E. Kratochvil, D.J. Bell, B.J. Nelson: Artificial bacterial flagella: Fabrication and magnetic control, Appl. Phys. Lett. **94**, 064107 (2009)

27.135 L. Zhang, E. Deckhardt, A. Weber, C. Schonenberger, D. Grutzmacher: Controllable fabrication of SiGe/Si and SiGe/Si/Cr helical nanobelts, Nanotechnology **16**, 655–663 (2005)

27.136 V.Y. Prinz, V.A. Seleznev, A.K. Gutakovsky, A.V. Chehovskiy, V.V. Preobrazhenskii, M.A. Putyato, T.A. Gavrilova: Free-standing and overgrown InGaAs/GaAs nanotubes, nanohelices and their arrays, Physica E Low-Dimen. Syst. Nanostructures **6**, 828–831 (2000)

Avidbots Ex1016
Page 745 of 2154

27.137 F.B. Hagedorn, E.M. Gyorgy: Magnetic-Shape Anisotropy in Polygonal Prisms, J. Appl. Phys. **39**, 995–997 (1968)

27.138 H.C. Berg, D.A. Brown: Chemotaxis in escherichia-coli analyzed by 3-dimensional tracking, Nature **239**, 500–504 (1972)

27.139 A. Ashkin, J.M. Dziedzic: Optical trapping and manipulation of viruses and bacteria, Science **235**, 1517–1520 (1987)

27.140 F.H.C. Crick, A.F.W. Hughes: The physical properties of cytoplasm: A study by means of the magnetic particle method, Part I, Exp. Cell Res. **1**, 37–80 (1950)

27.141 M.F. Yu, M.J. Dyer, G.D. Skidmore, H.W. Rohrs, X.K. Lu, K.D. Ausman, J.R.V. Ehr, R.S. Ruoff: Three-dimensional manipulation of carbon nanotubes under a scanning electron microscope, Nanotechnology **10**, 244–252 (1999)

27.142 L.X. Dong, F. Arai, T. Fukuda: 3D nanorobotic manipulation of nano-order objects inside SEM, Proc. Int. Symp. Micromechatron. Hum. Sci. (MHS) (2000) pp. 151–156

27.143 D.M. Eigler, E.K. Schweizer: Positioning single atoms with a scanning tunneling microscope, Nature **344**, 524–526 (1990)

27.144 P. Avouris: Manipulation of matter at the atomic and molecular levels, Acc. Chem. Res. **28**, 95–102 (1995)

27.145 M.F. Crommie, C.P. Lutz, D.M. Eigler: Confinement of electrons to quantum corrals on a metal surface, Science **262**, 218–220 (1993)

27.146 L.J. Whitman, J.A. Stroscio, R.A. Dragoset, R.J. Cellota: Manipulation of adsorbed atoms and creation of new structures on room-temperature surfaces with a scanning tunneling microscope, Science **251**, 1206–1210 (1991)

27.147 I.-W. Lyo, P. Avouris: Field-induced nanometer-scale to atomic-scale manipulation of silicon surfaces with the STM, Science **253**, 173–176 (1991)

27.148 G. Dujardin, R.E. Walkup, P. Avouris: Dissociation of individual molecules with electrons from the tip of a scanning tunneling microscope, Science **255**, 1232–1235 (1992)

27.149 T.-C. Shen, C. Wang, G.C. Abeln, J.R. Tucker, J.W. Lyding, P. Avouris, R.E. Walkup: Atomic-scale desorption through electronic and vibrational-excitation mechanisms, Science **268**, 1590–1592 (1995)

27.150 M.T. Cuberes, R.R. Schittler, J.K. Gimzewsk: Room-temperature repositioning of individual C60 molecules at Cu steps: operation of a molecular counting device, Appl. Phys. Lett. **69**, 3016–3018 (1996)

27.151 H.J. Lee, W. Ho: Single-bond formation and characterization with a scanning tunneling microscope, Science **286**, 1719–1722 (1999)

27.152 T. Yamamoto, O. Kurosawa, H. Kabata, N. Shimamoto, M. Washizu: Molecular surgery of DNA based on electrostatic micromanipulation, IEEE Trans. Ind. Appl. **36**, 1010–1017 (2000)

27.153 C. Haber, D. Wirtz: Magnetic tweezers for DNA micromanipulation, Rev. Sci. Instrum. **71**, 4561–4570 (2000)

27.154 D.M. Schaefer, R. Reifenberger, A. Patil, R.P. Andres: Fabrication of two-dimensional arrays of nanometer-size clusters with the atomic force microscope, Appl. Phys. Lett. **66**, 1012–1014 (1995)

27.155 T. Junno, K. Deppert, L. Montelius, L. Samuelson: Controlled manipulation of nanoparticles with an atomic force microscope, Appl. Phys. Lett. **66**, 3627–3629 (1995)

27.156 P.E. Sheehan, C.M. Lieber: Nanomachining, manipulation and fabrication by force microscopy, Nanotechnology **7**, 236–240 (1996)

27.157 C. Baur, B.C. Gazen, B. Koel, T.R. Ramachandran, A.A.G. Requicha, L. Zini: Robotic nanomanipulation with a scanning probe microscope in a networked computing environment, J. Vac. Sci. Tech. B **15**, 1577–1580 (1997)

27.158 A.A.G. Requicha: Nanorobots, NEMS, and nanoassembly, Proceedings IEEE **91**, 1922–1933 (2003)

27.159 R. Resch, C. Baur, A. Bugacov, B.E. Koel, A. Madhukar, A.A.G. Requicha, P. Will: Building and manipulating 3-D and linked 2-D structures of nanoparticles using scanning force microscopy, Langmuir **14**, 6613–6616 (1998)

27.160 J. Hu, Z.-H. Zhang, Z.-Q. Ouyang, S.-F. Chen, M.-Q. Li, F.-J. Yang: Stretch and align virus in nanometer scale on an atomically flat surface, J. Vac. Sci. Tech. B **16**, 2841–2843 (1998)

27.161 M. Sitti, S. Horiguchi, H. Hashimoto: Controlled pushing of nanoparticles: modeling and experiments, IEEE/ASME Trans. Mechatron. **5**, 199–211 (2000)

27.162 M. Guthold, M.R. Falvo, W.G. Matthews, S. Paulson, S. Washburn, D.A. Erie, R. Superfine, J.F.P. Brooks, I.R.M. Taylor: Controlled manipulation of molecular samples with the nanoManipulator, IEEE/ASME Trans. Mechatron. **5**, 189–198 (2000)

27.163 G.Y. Li, N. Xi, M.M. Yu, W.K. Fung: Development of augmented reality system for AFM-based nanomanipulation, IEEE/ASME Trans. Mechatron. **9**, 358–365 (2004)

27.164 F. Arai, D. Andou, T. Fukuda: Micro manipulation based on micro physics–strategy based on attractive force reduction and stress measurement, Proc. IEEE/RSJ Int. Conf. Intell. Robotics Syst. (1995) pp. 236–241

27.165 H.W.P. Koops, J. Kretz, M. Rudolph, M. Weber, G. Dahm, K.L. Lee: Characterization and application of materials grown by electron-beam-induced deposition, Jpn. J. Appl. Phys. **33**, 7099–7107 (1994)

27.166 S. Iijima: Helical microtubules of graphitic carbon, Nature **354**, 56–58 (1991)

27.167 S.J. Tans, A.R.M. Verchueren, C. Dekker: Room-temperature transistor based on a single carbon nanotube, Nature **393**, 49–52 (1998)

Avidbots Ex1016
Page 746 of 2154

27.168 R.H. Baughman, A.A. Zakhidov, W.A. de Heer: Carbon nanotubes–the route toward applications, Science **297**, 787–792 (2002)

27.169 M.J. Treacy, T.W. Ebbesen, J.M. Gibson: Exceptionally high Young's modulus observed for individual carbon nanotubes, Nature **381**, 678–680 (1996)

27.170 P. Poncharal, Z.L. Wang, D. Ugarte, W.A. de Heer: Electrostatic deflections and electromechanical resonances of carbon nanotubes, Science **283**, 1513–1516 (1999)

27.171 M.F. Yu, O. Lourie, M.J. Dyer, K. Moloni, T.F. Kelley, R.S. Ruoff: Strength and breaking mechanism of multiwalled carbon nanotubes under tensile load, Science **287**, 637–640 (2000)

27.172 T.W. Ebbesen, H.J. Lezec, H. Hiura, J.W. Bennett, H.F. Ghaemi, T. Thio: Electrical conductivity of individual carbon nanotubes, Nature **382**, 54–56 (1996)

27.173 P. Kim, L. Shi, A. Majumdar, P.L. McEuen: Thermal transport measurements of individual multiwalled nanotubes, Phys. Rev. Lett. **87**, 215502 (2001)

27.174 W.J. Liang, M. Bockrath, D. Bozovic, J.H. Hafner, M. Tinkham, H. Park: Fabry–Perot interference in a nanotube electron waveguide, Nature **411**, 665–669 (2001)

27.175 X.B. Zhang, D. Bernaerts, G.V. Tendeloo, S. Amelincks, J.V. Landuyt, V. Ivanov, J.B. Nagy, P. Lambin, A.A. Lucas: The texture of catalytically grown coil-shaped carbon nanotubules, Europhys. Lett. **27**, 141–146 (1994)

27.176 X.Y. Kong, Z.L. Wang: Spontaneous polarization-induced nanohelixes, nanosprings, and nanorings of piezoelectric nanobelts, Nano Lett. **3**, 1625–1631 (2003)

27.177 S.V. Golod, V.Y. Prinz, V.I. Mashanov, A.K. Gutakovsky: Fabrication of conducting GeSi/Si micro- and nanotubes and helical microcoils, Semicond. Sci. Technol. **16**, 181–185 (2001)

27.178 L. Zhang, E. Deckhardt, A. Weber, C. Schönenberger, D. Grützmacher: Controllable fabrication of SiGe/Si and SiGe/Si/Cr helical nanobelts, Nanotechnology **16**, 655–663 (2005)

27.179 L. Zhang, E. Ruh, D. Grützmacher, L.X. Dong, D.J. Bell, B.J. Nelson, C. Schönenberger: Anomalous coiling of SiGe/Si and SiGe/Si/Cr helical nanobelts, Nano Lett. **6**, 1311–1317 (2006)

27.180 D.J. Bell, L.X. Dong, B.J. Nelson, M. Golling, L. Zhang, D. Grützmacher: Fabrication and characterization of three-dimensional InGaAs/GaAs nanosprings, Nano Lett. **6**, 725–729 (2006)

27.181 D.J. Bell, Y. Sun, L. Zhang, L.X. Dong, B.J. Nelson, D. Grutzmacher: Three-dimensional nanosprings for electromechanical sensors, Sens. Actuators A Phys. **130**, 54–61 (2006)

27.182 R. Martel, T. Schmidt, H.R. Shea, T. Herte, P. Avouris: Single- and multi-wall carbon nanotube field-effect transistors, Appl. Phys. Lett. **73**, 2447–2449 (1998)

27.183 N.R. Franklin, Y.M. Li, R.J. Chen, A. Javey, H.J. Dai: Patterned growth of single-walled carbon nanotubes on full 4-inch wafers, Appl. Phys. Lett. **79**, 4571–4573 (2001)

27.184 T. Rueckes, K. Kim, E. Joselevich, G.Y. Tseng, C.-L. Cheung, C.M. Lieber: Carbon nanotube-based non-volatile random access memory for molecular computing science, Science **289**, 94–97 (2000)

27.185 A. Subramanian, B. Vikramaditya, L.X. Dong, D.J. Bell, B.J. Nelson: Micro and nanorobotic assembly using dielectrophoresis. In: *Robotics Sci. Syst*, ed. by S. Thrun, G.S. Sukhatme, S. Schaal, O. Brock (MIT Press, Cambridge 2005) pp. 327–334

27.186 C.K.M. Fung, V.T.S. Wong, R.H.M. Chan, W.J. Li: Dielectrophoretic batch fabrication of bundled carbon nanotube thermal sensors, IEEE Trans. Nanotechnol. **3**, 395–403 (2004)

27.187 T. Fukuda, F. Arai, L.X. Dong: Assembly of nanodevices with carbon nanotubes through nanorobotic manipulations, Proceedings IEEE **91**, 1803–1818 (2003)

27.188 E.W. Wong, P.E. Sheehan, C.M. Lieber: Nanobeam mechanics: elasticity, strength, and toughness of nanorods and nanotubes, Science **277**, 1971–1975 (1997)

27.189 M.R. Falvo, G.J. Clary, R.M. Taylor, V. Chi, F.P. Brooks, S. Washburn, R. Superfine: Bending and buckling of carbon nanotubes under large strain, Nature **389**, 582–584 (1997)

27.190 H.W.C. Postma, A. Sellmeijer, C. Dekker: Manipulation and imaging of individual single-walled carbon nanotubes with an atomic force microscope, Adv. Mater. **12**, 1299–1302 (2000)

27.191 T. Hertel, R. Martel, P. Avouris: Manipulation of individual carbon nanotubes and their interaction with surfaces, J. Phys. Chem. B **102**, 910–915 (1998)

27.192 P. Avouris, T. Hertel, R. Martel, T. Schmidt, H.R. Shea, R.E. Walkup: Carbon nanotubes: nanomechanics, manipulation, and electronic devices, Appl. Surf. Sci. **141**, 201–209 (1999)

27.193 M. Ahlskog, R. Tarkiainen, L. Roschier, P. Hakonen: Single-electron transistor made of two crossing multiwalled carbon nanotubes and its noise properties, Appl. Phys. Lett. **77**, 4037–4039 (2000)

27.194 M.R. Falvo, R.M.I. Taylor, A. Helser, V. Chi, F.P.J. Brooks, S. Washburn, R. Superfine: Nanometre-scale rolling and sliding of carbon nanotubes, Nature **397**, 236–238 (1999)

27.195 B. Bhushan, V.N. Koinkar: Nanoindentation hardness measurements using atomic-force microscopy, Appl. Phys. Lett. **64**, 1653–1655 (1994)

27.196 P. Vettiger, G. Cross, M. Despont, U. Drechsler, U. Durig, B. Gotsmann, W. Haberle, M.A. Lantz, H.E. Rothuizen, R. Stutz, G.K. Binnig: The *millipede* – Nanotechnology entering data storage, IEEE Trans. Nanotechnol. **1**, 39–55 (2002)

27.197 L.X. Dong: *Nanorobotic manipulations of carbon nanotubes. Ph.D. Thesis Ser* (Nagoya Univ., Nagoya 2003)

27.198 J.H. Hafner, C.-L. Cheung, T.H. Oosterkamp, C.M. Lieber: High-yield assembly of individual

single-walled carbon nanotube tips for scanning probe microscopies, J. Phys. Chem. B **105**, 743–746 (2001)
27.199 L.X. Dong, F. Arai, T. Fukuda: Electron-beam-induced deposition with carbon nanotube emitters, Appl. Phys. Lett. **81**, 1919–1921 (2002)
27.200 L.X. Dong, F. Arai, T. Fukuda: 3D nanorobotic manipulations of multi-walled carbon nanotubes, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (2001) pp. 632–637
27.201 L.X. Dong, F. Arai, T. Fukuda: Destructive constructions of nanostructures with carbon nanotubes through nanorobotic manipulation, IEEE/ASME Trans. Mechatron. **9**, 350–357 (2004)
27.202 J. Cumings, P.G. Collins, A. Zettl: Peeling and sharpening multiwall nanotubes, Nature **406**, 58 (2000)
27.203 J. Cumings, A. Zettl: Low-friction nanoscale linear bearing realized from multiwall carbon nanotubes, Science **289**, 602–604 (2000)
27.204 A. Kis, K. Jensen, S. Aloni, W. Mickelson, A. Zettl: Interlayer forces and ultralow sliding friction in multiwalled carbon nanotubes, Phys. Rev. Lett. **97**, 025501 (2006)
27.205 L.X. Dong, F. Arai, T. Fukuda: Nanoassembly of carbon nanotubes through mechanochemical nanorobotic manipulations, Jpn. J. Appl. Phys. **42**, 295–298 (2003)
27.206 L. Chico, V.H. Crespi, L.X. Benedict, S.G. Louie, M.L. Cohen: Pure carbon nanoscale devices: Nanotube heterojunctions, Phys. Rev. Lett. **76**, 971–974 (1996)
27.207 Z. Yao, H.W.C. Postma, L. Balents, C. Dekker: Carbon nanotube intramolecular junctions, Nature **402**, 273–276 (1999)
27.208 H.W.C. Postma, T. Teepen, Z. Yao, M. Grifoni, C. Dekker: Carbon nanotube single-electron transistors at room temperature, Science **293**, 76–79 (2001)
27.209 M.S. Fuhrer, J. Nygård, L. Shih, M. Forero, Y.-G. Yoon, M.S.C. Mazzoni, H.J. Choi, J. Ihm, S.G. Louie, A. Zettl, P.L. McEuen: Crossed nanotube junctions, Science **288**, 494–497 (2000)
27.210 T. Rueckes, K. Kim, E. Joselevich, G.Y. Tseng, C.-L. Cheung, C.M. Lieber: Carbon nanotube-based nonvolatile random access memory for molecular computing science, Science **289**, 94–97 (2000)
27.211 A.G. Rinzler, J.H. Hafner, P. Nikolaev, L. Lou, S.G. Kim, D. Tománek, P. Nordlander, D.T. Colbert, R.E. Smalley: Unraveling nanotubes: field emission from an atomic wire, Science **269**, 1550–1553 (1995)
27.212 S.C. Minne, G. Yaralioglu, S.R. Manalis, J.D. Adams, J. Zesch, A. Atalar, C.F. Quate: Automated parallel high-speed atomic force microscopy, Appl. Phys. Lett. **72**, 2340–2342 (1998)
27.213 G.D. Skidmore, E. Parker, M. Ellis, N. Sarkar, R. Merkle: Exponential assembly, Nanotechnology **11**, 316–321 (2001)
27.214 H.J. Dai, J.H. Hafner, A.G. Rinzler, D.T. Colbert, R.E. Smalley: Nanotubes as nanoprobes in scanning probe microscopy, Nature **384**, 147–150 (1996)
27.215 J.H. Hafner, C.L. Cheung, C.M. Lieber: Growth of nanotubes for probe microscopy tips, Nature **398**, 761–762 (1999)
27.216 L.X. Dong, B.J. Nelson, T. Fukuda, F. Arai: Towards Nanotube Linear Servomotors, IEEE Trans. Autom. Sci. Eng. **3**, 228–235 (2006)
27.217 Y.H. Gao, Y. Bando: Carbon nanothermometer containing gallium, Nature **415**, 599 (2002)
27.218 L.X. Dong, X.Y. Tao, L. Zhang, B.J. Nelson, X.B. Zhang: Nanorobotic spot welding: Controlled metal deposition with attogram precision from Copper-filled carbon nanotubes, Nano Lett. **7**, 58–63 (2007)
27.219 S.W. Lee, D.S. Lee, R.E. Morjan, S.H. Jhang, M. Sveningsson, O.A. Nerushev, Y.W. Park, E.E.B. Campbell: A three-terminal carbon nanorelay, Nano Lett. **4**, 2027–2030 (2004)
27.220 A.M. Fennimore, T.D. Yuzvinsky, W.-Q. Han, M.S. Fuhrer, J. Cumings, A. Zettl: Rotational actuators based on carbon nanotubes, Nature **424**, 408–410 (2003)
27.221 A. Subramanian, L.X. Dong, J. Tharian, U. Sennhauser, B.J. Nelson: Batch fabrication of carbon nanotube bearings, Nanotechnology **18**, 075703 (2007)
27.222 P. Kim, C.M. Lieber: Nanotube nanotweezers, Science **286**, 2148–2150 (1999)
27.223 L.X. Dong, A. Subramanian, D. Hugentobler, B.J. Nelson, Y. Sun: Nano Encoders based on Vertical Arrays of Individual Carbon Nanotubes, Adv. Robotics **20**, 1281–1301 (2006)
27.224 L.X. Dong, B.J. Nelson: Robotics in the small, Part II: Nanorobotics, IEEE Robotics Autom. Mag. **14**, 111–121 (2007)
27.225 I. Asimov: *Fantastic Voyage* (Bantam Books, New York 1966)
27.226 R.A. Freitas: *Nanomedicine, Volume I: Basic Capabilities* (Landes Bioscience, Austin 1999)

Avidbots Ex1016
Page 748 of 2154

Multimedia Contents

Part C Sensing and Perception

Ed. by Henrik I. Christensen

28 Force and Tactile Sensing
Mark R. Cutkosky, Stanford, USA
William Provancher, Salt Lake City, USA

29 Inertial Sensing, GPS and Odometry
Gregory Dudek, Montreal, Canada
Michael Jenkin, Toronto, Canada

30 Sonar Sensing
Lindsay Kleeman, Melbourne, Australia
Roman Kuc, New Haven, USA

31 Range Sensing
Kurt Konolige, Mountain View, USA
Andreas Nüchter, Würzburg, Germany

32 3-D Vision for Navigation and Grasping
Danica Kragic, Stockholm, Sweden
Kostas Daniilidis, Philadelphia, USA

33 Visual Object Class Recognition
Michael Stark, Saarbrücken, Germany
Bernt Schiele, Saarbrücken, Germany
Aleš Leonardis, Birmingham, UK

34 Visual Servoing
François Chaumette, Rennes, France
Seth Hutchinson, Urbana-Champaign, USA
Peter Corke, Brisbane, Australia

35 Multisensor Data Fusion
Hugh Durrant-Whyte, Sydney, Australia
Thomas C. Henderson, Salt Lake City, USA

Part C covers material related to sensing and perception. The section covers all the aspects of sensing for basic measurement of physical parameters in the world to making sense of such data for the purpose of enabling a robot to perform its tasks. Right now robotics is seeing a revolution in use of sensors. Traditionally robots have been designed to have maximum stiffness and applications have been designed to be predictable in their operation. As robots emerge from the fences areas and we deploy robots for a wider range of applications from collaborative robotics to autonomously driving cars it is essential to have perception capabilities that allow estimation of the state of the robot but also the state the surrounding environment. Due to these new requirements the importance of sensing and perception has increased significantly over the last decade and will without doubt to continue to grow in the future.

The topics addressed in this section include all aspects from detection and processing physical contact with the world through force and tackle sensing over augmented environments for detection of position and motion in the world to image based methods for mapping, detection and control in structured and unstructured environments. In many settings a single sensory modality/sensor is inadequate to give robust estimate of the state of the environment. Consequently a chapter on multi-sensory fusion is also included in this part. Part C considers primarily the sensing and perception aspects of robotics with a limited view to many of the other aspects. Part A is providing the basic, Part B provides the kinematic structures that we need to design control methods for interaction, while Part D will cover the grasping and manipulation of objects. The remaining parts of the handbook covers application verticals where sensors and perception plays a key role.

With this brief overview of Part C, we provide a brief synopsis of each chapter.

Chapter 28 covers force and tactile sensing, which is essential for physical interaction with the world. Tactile sensing addresses the issue of detecting and handling contact with objects and structures in the world. Tactile sensing is essential both for safety and manipulation applications as the control of a robot changes due to contact which changes the kinematic configurations. Force sensing addresses estimation of force and torque as part of dynamic motion but also for interaction with physical objects such as inserting an object or turning a valve. The model chosen for force estimation also impacts the control of the robot.

Chapter 29 covers use of odometry, inertial and GPS for estimation of motion and position in the world. Odometry is the ego-estimation of your position/motion based on robot mounted sensors. With the introduction of sensors for estimation of accelerator and rotational velocity it is possible improve the estimation of position and motion. With the introduction of inexpensive inertial measurement units (IMU), driven by the game and phone industry, the use of IMU has increased significantly. For outdoor operation it is frequently possible to utilize Global Navigation Systems such as the global positioning systems (GPS). The integration of odometry, IMU and GPS information allow design of systems for precision agriculture, autonomous driving, etc. The chapter provides a description of the core techniques to make this possible.

One of the first sensors to be used for large scale estimation of distance to external objects was sonars. Chapter 30 covers the basic methods for sound based ranging and localization. Sonars are widely used for underwater mapping and localization, but is also used as an inexpensive modality for localization and ground vehicles and landing of Unmanned Aerial Vehicles. Typically a single sensor has limited precision but through used of phased array techniques and multiple sensors it is possible to achieve high-fidelity ranging. The chapter covers both the basic physics, sensing, and estimation methods.

Over the last two decades we have seen tremendous progress on laser based ranging as a complement to stereo/multi-ocular methods for distance estimation. Light detection and ranging (LIDAR) is today used as an effective modality for mapping, localization and tracking of external objects in the environment. In Chapter 31 the basic techniques for range estimation using light based ranging is described and fundamental methods for three-dimensional (3-D) modeling of the environment are described. Recently there has been a renaissance in range based sensing due to the introduction of RGB-D sensors, that utilize structured light and cameras to generate a dense range maps at very-low prices.

Chapter 32 covers the general topics of 3-D vision which covers estimation of distance from the parallax between two or more cameras and/or the motion of a camera over time. The parallax between two or more images allows estimation of the distance to external objects, which can be leveraged for localization and navigation, but also for grasping and interacting with objects in the environment. The chapter covers both the basic aspects of 3-D estimation and some of the common applications of 3-D vision for robot control.

Chapter 33 covers the topic of object recognition. Computer based object recognition has been studied for more than 50 years. Over the last decade we have

seen tremendous progress on object recognition due to improved cameras and availability of much better computers and increased memory. We are seeing both image based recognition techniques and methods for recognition based on the 3-D structure of objects. More recently we have seen a renewed interesting use of neural nets for detection of objects due to new Bayesian methods and the problem of object categorization – recognizing categories of objects such as car, motorcycle, traffic sign, people, etc. has become an important problem. The chapter covers both view and 3-D based methods for recognition and discusses also categorization of objects.

Chapter 34 covers the problem of image servoing. When trying to interact with an object we can use image data to drive the end-effector to a goal location. Two common configurations are eye-in-hand and hand-to-eye. In addition the control can be performed directly in image coordinates or through recovery of two-and-a-half-dimensional (2.5-D) or 3-D pose for an object. To enable all of this an impact aspect is to derived the relation between changes in robot motion and changes in the image/pose, which is derivation of the Jacobian for the system. This chapter discusses both eye-in-hand and hand-to-eye visual servoing and image/pose based control of the process and provides examples of use of visual servoing in real scenarios.

Finally, the part is concluded by Chapter 35 which discusses multi-sensory data fusion. As mentioned earlier most applications require use of multiple sensors to generate robust/complete methods for control. Data fusion involves a number of different aspects from time synchronization to transformation into a common reference frame to integration of data over time/space to generate more robust/accurate estimate of the world state. Recently a number of new methods for Bayesian data fusion have emerged. In this chapter the fundamental of data fusion are reviewed and a number of the most common techniques for multi-sensory fusion are introduced.

Multimedia Contents

28. Force and Tactile Sensing

Mark R. Cutkosky, William Provancher

This chapter provides an overview of force and tactile sensing, with the primary emphasis placed on tactile sensing. We begin by presenting some basic considerations in choosing a tactile sensor and then review a wide variety of sensor types, including proximity, kinematic, force, dynamic, contact, skin deflection, thermal, and pressure sensors. We also review various transduction methods, appropriate for each general sensor type. We consider the information that these various types of sensors provide in terms of whether they are most useful for manipulation, surface exploration or being responsive to contacts from external agents.

Concerning the interpretation of tactile information, we describe the general problems and present two short illustrative examples. The first involves intrinsic tactile sensing, i. e., estimating contact locations and forces from force sensors. The second involves contact pressure sensing, i. e., estimating surface normal and shear stress distributions from an array of sensors in an elastic skin. We conclude with a brief discussion of the challenges that remain to be solved in packaging and manufacturing damage-tolerant tactile sensors.

28.1 **Overview** 717
28.2 **Sensor Types** 718
28.2.1 Proprioceptive and Proximity Sensing 718
28.2.2 Other Contact Sensors 720
28.2.3 Kinematic Sensors 721
28.2.4 Force and Load Sensing 721
28.2.5 Dynamic Tactile Sensors 722
28.2.6 Array Sensors 722
28.3 **Tactile Information Processing** 725
28.3.1 Tactile Information Flow: Means and Ends of Tactile Sensing 725
28.3.2 Solid Mechanics and Deconvolution 727
28.3.3 Curvature and Shape Information 729
28.3.4 Object and Surface Identification 729
28.3.5 Active Sensing Strategies 730
28.3.6 Dynamic Sensing and Event Detection 730
28.3.7 Integration of Thermal and Other Sensors 730
28.4 **Integration Challenges** 730
28.5 **Conclusions and Future Developments** .. 731
Video-References 731
References 731

28.1 Overview

Tactile sensing has been a component of robotics for roughly as long as vision. However, in comparison to vision, for which great strides have been made in terms of hardware and software and which is now widely used in industrial and mobile robot applications, tactile sensing always seems to be *a few years away* from widespread utility. So before reviewing the technologies and approaches available, it is worthwhile to ask some basic questions:

- How important is tactile sensing?
- What is it useful for?
- Why does it remain comparatively undeveloped?

In nature, tactile sensing is an essential survival tool. Even the simplest creatures are endowed with large numbers of mechanoreceptors for exploring and responding to various stimuli. In humans, tactile sensing is indispensable for three distinct kinds of activity: ma-

nipulation, exploration and response. The importance of tactile sensing for manipulation is most evident in fine motor tasks. When we are chilled, tasks like buttoning a shirt can become an exercise in frustration. The problem is primarily a lack of sensing; our muscles, snug in our coat sleeves, are only slightly affected but our cutaneous mechanoreceptors are anesthetized and we become clumsy. For exploration, we continually assimilate information about materials and surface properties (e.g., hardness, thermal conductivity, friction, roughness) to help us identify objects. We may have difficulty distinguishing real leather from synthetic leather by sight, but not by touch. Finally, the importance of tactile response, whether to a gentle touch or an impact, is seen in the damage that patients with peripheral neuropathy (e.g., as a complication of diabetes) accidentally do to themselves.

As Fig. 28.1 indicates, the same functional categories apply to robots. However, in comparison to animals, with hundreds or thousands of mechanoreceptors per square centimeter of skin, even the most sophisticated robots are impoverished. One reason for the slow development of tactile sensing technology as compared to vision is that there is no tactile analog to the charge-coupled device (CCD) or complementary metal-oxide semiconductor (CMOS) optical array. Instead, tactile sensors elicit information through physical interaction. They must be incorporated into skin surfaces with compliance, for conforming locally to surfaces, and with adequate friction for handling objects securely. The sensors and skin must also be robust enough to survive repeated impacts and abrasions. And unlike the image plane in a camera, tactile sensors must

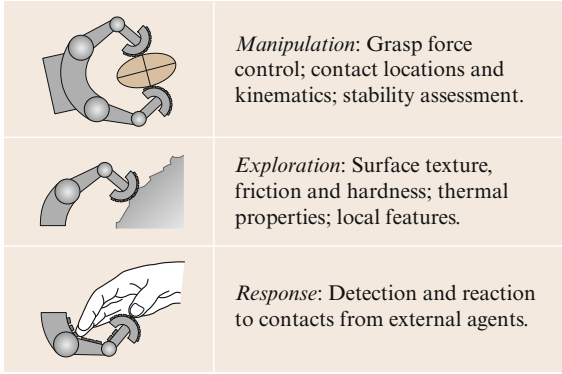


Fig. 28.1 Uses of tactile sensing in robotics

be distributed over the robot appendages, with particularly high concentrations in areas like the fingertips. The wiring of tactile sensors is consequently another formidable challenge.

Nonetheless, considerable progress in tactile sensor design and deployment has been made over the last couple of decades. In the following sections we review the main functional classes of tactile sensors and discuss their relative strengths and limitations. Looking ahead, new fabrication techniques offer the possibility of artificial skin materials with integrated sensors and local processing for interpreting sensor signals and communicating over a common bus to reduce wiring.

There is an extensive literature describing touch sensing research. Recent general reviews include [28.1–4] and these cite a number of useful older reviews including [28.5–7].

28.2 Sensor Types

This section outlines five main types of sensors: proprioceptive, kinematic, force, dynamic tactile, and array tactile sensors, as well as sensors that provide thermal or material composition data. However, the emphasis is on sensors that provide mechanoreception, as summarized in Table 28.1. The most important quantities measured with tactile sensors are *shape* and *force*. Each of these may be measured as an average quantity for some part of the robot or as a spatially resolved, distributed quantity across a contact area. In this chapter we follow the convention of studies of the human sense of touch and use the term *touch sensing* to refer to the combination of these two modes. Devices that measure an average or resultant quantity are sometimes referred to as *internal* or *intrinsic* sensors. The basis for these sensors is force sensing, which precedes the discussion of tactile array sensors.

28.2.1 Proprioceptive and Proximity Sensing

Proprioceptive sensing refers to sensors that provide information about the net force or motion of an appendage, analogous to receptors that provide information in humans about tendon tensions or joint movements. Generally speaking, the primary source for spatial proprioceptive information on a robot is provided by joint angle and force–torque sensors. Since joint angle sensors such as potentiometers, encoders, and resolvers are well established technologies, they do not warrant discussion here. Instead, a brief review of proximity sensing via whiskers and antennae as well as noncontact proximity sensing is provided. Force–torque sensors are discussed in greater detail in Sect. 28.2.4.

Avidbots Ex1016
Page 753 of 2154

Table 28.1 Tactile sensor modalities and common transduction types

Sensor modality	Sensor type	Sensor attributes	Advantages	Disadvantages
Normal pressure	Piezoresistive array [28.8–12]	Array of piezoresistive junctions Embedded in a elastomeric skin Cast or screen printed	Suitable for mass production Simple design Simple signal conditioning	Low force thresholds/high sensitivity Frail Signal drift and hysteresis
	Capacitive array [28.13–17]	Array of capacitive junctions Row and column electrodes separated by elastomeric dielectric	Good sensitivity Moderate hysteresis, depending on construction	Complex circuitry
	Piezoresistive microelectromechanical system (MEMS) array [28.18–20]	Silicon micromachined array with doped silicon strain gauged flexures	Suitable for to mass production	Frail
	Optical [28.21, 22]	Combined tracking of optical markers with a constitutive model	No interconnects to break	Requires personal computer (PC) for computing applied forces
Skin deformation	Optical [28.23, 24]	Fluid-filled elastomeric membrane Tracking of optical markers inscribed on membrane coupled with energy minimization algorithm	Compliant membrane No electrical interconnects to be damaged	Complex computations Hard to customize sensor
	Magnetic [28.25]	Array of hall effect sensors		Complex computations Hard to customize sensor
	Resistive tomography [28.26]	Array of conductive rubber traces as electrodes	Robust construction	
	Piezoresistive (curvature) [28.27, 28]	Employs an array of strain gauges	Directly measure curvature	Frailty of electrical interconnects Hysteresis
Dynamic tactile sensing	Piezoelectric (stress rate) [28.19, 29, 30]	polyvinylidene fluoride (PVDF) embedded in elastomeric skin	High bandwidth	Frailty of electrical junctions
	Skin acceleration [28.31, 32]	Commercial accelerometer afixed to robot skin	Simple	No spatially distributed content Sensed vibrations tend to be dominated by structural resonant frequency

Avidbots Ex1016
Page 754 of 2154

Whisker and Antenna Sensors

For many animals whiskers or antennae provide an extremely accurate combination of contact sensing and proprioceptive information. For example, cockroaches can steer themselves along curved walls using only position and rate information from their antennae [28.33]. Other insects and arthropods use numerous small hair sensors on the exoskeleton to localize contacts. Rats perform highly accurate whisking to explore the shapes and textures of objects in their vicinity, with sensor processing performed by a specialized barrel cortex [28.34].

In robotics, this potentially very useful hybrid of proprioceptive and tactile sensing has received comparatively little attention, although examples date at least to the early 1990s [28.35, 36]. In more recent work, *Clements* and *Rahn* [28.37] demonstrated an active whisker with a sweeping motion and *Lee* et al. [28.33] used a passive, flexible antenna to steer a running cockroach-inspired robot. *Prescott* et al. [28.34, 38] have conducted extensive work on active robotic whisking for object exploration and identification, inspired by mammalian models.

Proximity

While proximity sensing does not strictly fall under the category of tactile sensing, a number of researchers have employed proximity sensors for collision detection between a robot arm and the environment; hence, we briefly review these technologies here. Three primary sensor technologies which include capaciflective, infrared (IR) optical, and ultrasonic sensors have been used in this application. *Vranish* et al. developed an early capaciflective sensor for collision avoidance between the environment and a grounded robot arm [28.39]. Early examples of IR emitter/detector pairs in an artificial skin include [28.40, 41]. A more recent design using optical fibers is reported in [28.42], and an adaptation for a prosthetic hand is reported in [28.43]. Other researchers have developed robot skin that includes ultrasonic and IR optical sensors for collision avoidance [28.44]. *Wegerif* and *Rosinski* provide a comparison of the performance of all 3 of these proximity sensing technologies [28.45]. For a further review of some of these sensors, see Chap. 31 on range sensing.

28.2.2 Other Contact Sensors

There are a variety of other contact-based sensors that are capable of discerning object properties such as electromagnetic characteristics, density (via ultrasound), or chemical composition (cf. animals' senses of taste and smell). While this is beyond the scope of the current chapter, Chap. 75, on biologically inspired robots, briefly discusses biologically inspired chemical sensors related to smell and taste. For completeness, thermal sensors and material composition sensors are also briefly discussed below.

Thermal Sensors

Thermal sensing is an important element of human tactile sensing, useful for determining the material composition of an object as well as to measure surface temperatures. Since most objects in the environment are at about the same (*room*) temperature, a temperature sensor that contains a heat source can detect the rate at which heat is absorbed by an object. This provides information about the heat capacity of the object and the thermal conductivity of the material from which it is made, making it easy, for example, to distinguish metals from plastics.

Buttazzo et al. [28.46] note that the piezoelectric polymer used in their tactile sensing system is also strongly pyroelectric, and use a superficial layer as a thermal sensor. Other sensors use thermistors as transducers [28.47–49]. Some systems purposely provide an internal temperature reference and use the temperature differential from the environment to detect contacts [28.50, 51]; however, objects with a temperature the same as the reference will not be detected. Most of these sensors have a relatively thick outer skin covering the heat sensitive elements, thus protecting delicate components and providing a conformal surface at the expense of slower response time.

As a more recent example of thermal sensing, *Engel* et al. [28.52] present a flexible tactile sensor design that includes integrated gold film heaters and RTDs (resistance temperature devices) on a polymer micromachined substrate. *Lin* et al. [28.24] include a thermistor as part of the sensing suite in their artificial fingertip. While there is a high level of integration in these sensors, tradeoffs concerning construction, performance, and protection of sensing elements remain an ongoing challenge.

Material Composition Sensors

There has been some work on sensors for material composition. In analogy with the human senses of taste and smell, liquid and vapor phase chemical sensors could potentially determine the chemical composition of a surface [28.53, 54]. However, the large majority of robotic chemical sensing has involved non-contact sensing of airborne plumes [28.55, 56]. Another sensing modality which provides information about material properties is electromagnetic field sensing, using devices such as eddy current or Hall effect probes to measure ferromagnetism or conductivity [28.57, 58].

28.2.3 Kinematic Sensors

Although they are not generally regarded as tactile sensors, sensors that detect the position of a limb can provide the robot with geometric information for manipulation and exploration, particularly when the limb also includes sensors that register contact events. Examples of combining joint angle sensing and contact sensing with compliant fingers to learn about the configuration of a grasp include [28.59, 60].

28.2.4 Force and Load Sensing

Actuator Effort Sensors

For some actuators such as electric servo motors, a measure of the actuator effort can be obtained directly by measuring the motor current (typically using a sensing resistor in series with the motor). However, because motors are typically connected to robot limbs via gearboxes with output/input efficiencies of 60% or less, it is usually much more accurate to measure the torque at the output of the gearbox. Solutions to this problem include torque load cells and mechanical structures at the robot joints whose deflections can be measured using electromagnetic or optical sensors. For cable or tendon-driven arms and hands it is useful to measure the cable tension – both for purposes of compensating for friction in the drive-train and as a way of measuring the loads upon the appendage [28.61, 62]. When fingers or arms make contact with objects in the environment, cable tension sensing becomes an alternative to endpoint load sensing for measuring components of the contact forces. Of course, only those components that produce significant torques can be measured with accuracy. Chapter 19 on multifingered hands contains more details concerning tendon tension measurement.

Force Sensors

When actuator effort sensors are not sufficient to measure the forces exerted by or on a robot appendage, discrete force sensors are typically utilized. These sensors are found most often at the base joint or wrist of a robot but could be distributed throughout the links of a robot.

In principle, any type of multi-axis load cell could be used for manipulator force–torque sensing. However, the need for small, lightweight units with good static response eliminates many commercial sensors. The design of force sensors for mounting above the gripper at the wrist has received the most attention [28.63, 64], but fingertip sensors for dextrous hands have also been devised. Often these sensors are based on strain gauges mounted on a metal flexure [28.65–67] which can be fairly stiff and robust. *Sinden* and *Boie* [28.68] propose a planar six axis force–torque sensor based on capacitive measurements with an elastomer dielectric. Design considerations for force sensors include stiffness, hysteresis, calibration, amplification, robustness, and mounting. *Dario* et al. present an integrated fingertip for robotic hands integrated FSR (force sensing resistor) pressure array, piezoceramic bimorph dynamic sensor, and force–torque sensor [28.29]. More recently *Edin* et al. [28.69] have developed a miniature a multi-axis fingertip force sensor (Fig. 28.2). For applications where immunity to electromagnetic noise is desirable, *Park* et al. [28.70] present a robot fingertip with embedded fiber optic Bragg gratings, used as optical strain gages. *Bicchi* [28.71] and *Uchiyama* et al. [28.72] consider the optimal design of multi-axis force sensors in general.

Information from the force sensors can also be combined with knowledge of fingertip geometry to estimate contact location, as implied in Fig. 28.3. This method of contact sensing is referred to as *intrinsic tactile sensing* and was first presented by *Bicchi* et al. [28.73]. A comparison between intrinsic and extrinsic contact sensing

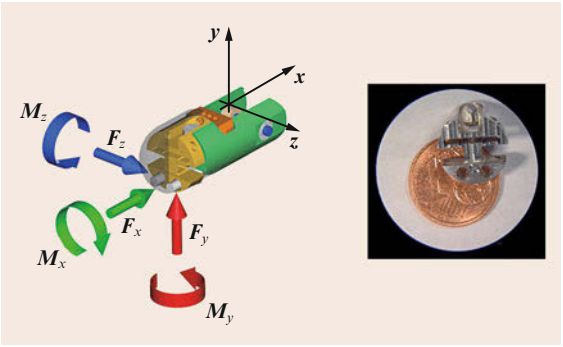


Fig. 28.2 Miniature fingertip force–torque sensor for a prosthetic hand (after [28.69])

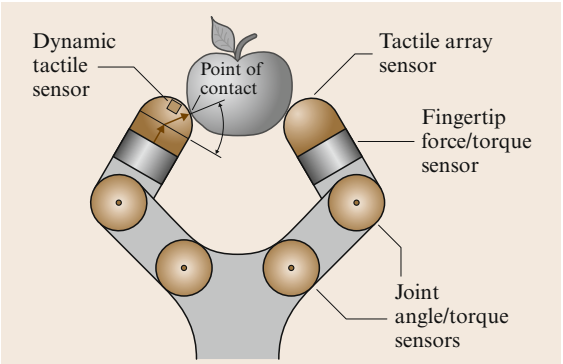


Fig. 28.3 Robot hand with fingertip force and tactile sensing. Information from the force sensors can be combined with knowledge of fingertip geometry to estimate contact location, referred to as *intrinsic tactile sensing*

(i. e., using distributed contact sensors) is presented by *Son* et al. [28.74]. This topic is discussed further in Sect. 28.3.1.

28.2.5 Dynamic Tactile Sensors

Taking a cue from the role of fast-acting or dynamic tactile sensors in human manipulation [28.75], researchers have developed dynamic sensors for slip detection and for sensing textures and fine features.

Early slip sensors based on displacement detected the motion of a moving element such as a roller or needle in the gripper surface [28.76, 77]. Subsequent work has typically used accelerometers or other sensors that are inherently sensitive to small, transient forces or motions. Early examples include [28.31, 78–80]. Many subsequent contributions are reviewed in [28.81].

For the case of hard objects held in metal grippers, acoustic emissions may reveal slip [28.82]. Because these signals are at very high frequency (over 100 kHz) they can be useful for distinguishing among different kinds of contact events in noisy environments [28.83]. Another approach for improving the signal/noise robustness of dynamic sensing is to use an actively stimulated sensor and measure the change in response as contact conditions change [28.84, 85].

For grasp force control, it is especially useful to detect incipient slips, which are accompanied by small, localized vibrations, or micro-slips, at the periphery of a finger/object contact region before gross sliding occurs. The challenge is to distinguish these from other events that produce vibrations [28.86–89].

For detecting features and fine surface features it can be effective to use small fibers, a skin with fingerprint-like ridges, or a stylus that drags over the surface like a fingernail [28.46, 90, 91].

Because many dynamic tactile sensors only produce a transient response, they are often combined with pressure sensing arrays or force sensors to provide a combination of low frequency and high frequency tactile sensing [28.24, 29, 32, 46, 89, 92, 93]. An alternative, which can be simpler from the standpoint of sensor integration, is to use conventional pressure sensing arrays for slip detection. In this case, the array resolution and scanning rate must be sufficient to detect motion or incipient motion quickly, to prevent grasp failures. Fortunately, this is becoming increasingly feasible [28.19, 81, 94, 95]. VIDEO 14 shows an example of dynamic tactile sensing.

28.2.6 Array Sensors

Hundreds of designs for tactile array sensors have appeared in the literature in the last 25 years, and many of them are designed for use with dextrous hands. In terms of transducers, the fundamental requirement is to unambiguously recover either the shape or pressure distribution across the contact. Shape sensing requires a compliant skin, which can also have advantages for grasp stability (see Chaps. 37 and 38 on contact modeling and grasping). Examples of shape-sensing tactile arrays include [28.23, 24]. However, the far more common approach is to measure subsurface strains, which can be correlated with surface pressure distributions, as presented by *Fearing* and *Hollerbach* [28.96] and discussed further in Sect. 28.3.2.

Contact Location Sensors

The simplest and perhaps most robust tactile arrays provide measurements solely of contact location. Some such sensors utilize a membrane switch design like that found in keyboards [28.97]. As another example, a robust two-dimensional (2-D) switch array can be embedded in a prosthetic hand [28.69]. Some optical tactile sensors have also been used primarily as contact location sensors. *Maekawa* et al. [28.98] used a single optical position sensing device (PSD) or a CCD camera array to detect the position of scattered light off of a hemispherical optical wave guide fingertip with a silicone rubber cover. Light is scattered at the locations of contacts. With a textured skin, the magnitude of the force can also be estimated, as the contact area grows in proportion to the pressure. However, an issue with fingertips that use a compliant skin covering a hard substrate is that adhesion between the two materials results in hysteresis. In addition, when the fingertip is dragged over a surface, the friction can produce a shift in the estimated contact position.

Pressure Sensing Arrays

Capacitive Pressure Sensing Arrays. Capacitive pressure sensing arrays are one of the oldest and most common tactile sensor types. Some of the earliest analysis of such sensors is presented by *Fearing* [28.99] for capacitive arrays embedded into a soft robot fingertip. The arrays consist of overlapping row and column electrodes separated by an elastomeric dielectric, forming an array of capacitors. A change in capacitance results from compressing the dielectric between row-column plates at a particular intersection. The equation for capacitance, based on the physical parameters, is expressed as $C \approx \epsilon A/d$, where ϵ is the permittivity of the dielectric between the plates of the capacitor, A is the area of the plates, and d is the spacing between them. Pressing on the skin reduces the plate spacing d, thus providing a linear response with displacement. Through appropriate switching circuitry, a region of a sensor array can be isolated at a particular

row/column intersection. Examples of similar capacitive tactile arrays can be found in [28.100, 101] and commercially [28.14]. Large capacitive arrays have also been developed to cover the arms of a robot [28.15–17]. A woven fabric with capacitive sensing junctions is reported in [28.102].

With a suitable dielectric and plate design, capacitive arrays can be robust, have a large dynamic range (ratio between minimum and maximum detectable pressure) and low hysteresis, which is desirable for fast response. However, a common problem is the need to shield the array from stray capacitance effects (e.g., from an approaching metallic surface) and to minimize *parasitic* capacitance associated with wiring to and from the active elements. For these reasons, the recent trend is to use local microprocessors specialized for capacitance measurement as part of an integrated sensing system (Sect. 28.4). An example is shown in Fig. 28.4. VIDEO 15 actually shows results taken with the sensor depicted in Fig. 28.4.

Piezoresistive Pressure Sensing Arrays. Many researchers have produced tactile sensor arrays that are piezoresistive. In most cases, these sensors utilize a conductive rubber that is bulk molded or a piezoresistive ink that is patterned via screen printing or stamping. Each of these approaches employs a conductive additive (typically carbon black or silver) to create its conductive/piezoresistive behavior. As a more flexible and durable alternative, some researchers have developed fabric-based piezoresistive sensors, discussed separately below.

Russell [28.27] presented one of the first molded conductive rubber tactile sensor arrays composed of conductive rubber column and row electrodes with piezoresistive junctions. However, this sensor exhibited significant drift and hysteresis, which fueled research to minimize these effects through proper selection of molding material [28.8]. These issues were never completely solved due to the hysteretic nature of elastomers, but this sensing approach remains attractive due to its ease of manufacturing. Hence, it has continued to find applications, for example, in the appendages of humanoid robots where extreme accuracy is not required [28.104]. Significant improvements in dynamic range and robustness are also possible using a treated elastomer that exploits a quantum tunneling effect [28.105], and commercial versions are now available [28.106].

A number of researchers and companies have developed tactile sensors that utilize conductive (piezoresistive) ink, generally referred to as FSRs (force sensitive resistors). This is by far the most common and economical means to incorporate tactile sensing via off-the-shelf sensors. However, to make highly integrated, dense sensor arrays, custom fabrication is necessary. Examples of such sensors include [28.9, 29]. To take this basic concept a step further, *Someya* [28.107] has produced robotic skin that employs patterned organic semiconductors for local amplification of the piezoresistive sensor array, printed on flexible polyimide film. Although fabricated on a flexible substrate, these sensor arrays are somewhat susceptible to bending fatigue.

Piezoresistive fabrics have been developed to address fatigue and fragility issues found in tactile arrays. Examples of these sensors are presented in [28.10–12, 108] and are utilized in applications such as the arms or legs of humanoid robots. Because this technology has the potential to replace ordinary cloth, it is a promising technology for applications in wearable computing or even smart clothing. A different solution to the problem of making a sensor that will work on joints and locations with extensive flexing is to employ fine channels of liquid metal in a stretchy elastomer [28.109].

One final design that does not fall under the above fabrication categories is a sensor designed by *Kageyama* et al. [28.110] that utilizes a piezoresistive conductive gel pressure array along with a multi-level contact switch array via variable contact resistance within sensor layers that they developed for use in humanoid robots.

MEMS arrays. MEMS technology is attractive for producing dense sensor arrays with integrated packaging, signal processing and communications electronics. Early devices were produced in silicon through standard silicon micro-machining techniques, allowing high spatial resolution and dedicated hardware for multi-

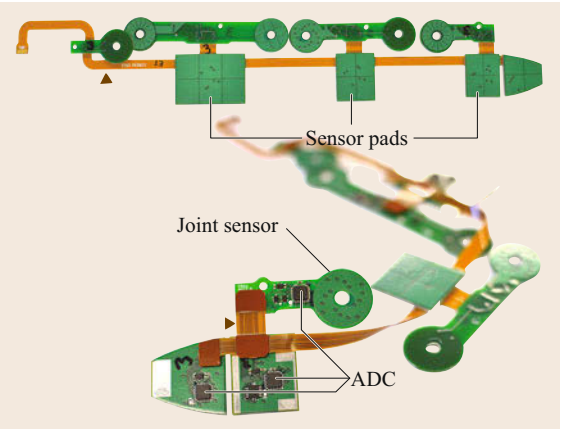


Fig. 28.4 Capacitive touch and joint angle sensors on a flexible circuit for incorporation in a robotic hand, sharing the same microprocessors for signal processing and communication (after [28.103])

plexing, etc. [28.111, 112]. However, such silicon-only devices can be brittle and difficult to integrate into a robust, compliant skin. More recent efforts have addressed these problems by using combinations of semiconductors and other materials, including organic semiconductors [28.18, 19, 52, 113, 114].

Returning to the idea of using silicon load cells for tactile sensing, *Valdastri* et al. [28.115] developed a miniature MEMS silicon-based three-axis load cell that resembles a joystick and is appropriate for embedding within an elastomeric skin for detecting shear and normal stresses (Fig. 28.5) [28.93]. Other recent efforts include flexible arrays with high spatial resolution using capacitive or piezoresistive technology [28.20, 116].

Skin Deflection Sensing

Brocket [28.117] was one of the first to propose the idea of using deformable membranes as robot fingertips. As noted by *Shimoga* and *Goldenberg* [28.118], there are several advantages to using deformable fingertips over more rigid robot fingertips, which include: 1) improved grasp stability, 2) reduced shock, and 3) reduced fatigue for embedded sensor elements. Other early work includes a deformable fingertip with skin covering polyurethane foam and instrumented with elastomeric strain gages [28.27].

Nowlin [28.25] used Bayesian algorithms to improve data interpretation of a deformable tactile sensor that used magnetic field sensing. A 4×4 array of magnets were supported above paired hall-effect sensors off a rigid base by individual fluid filled balloons. Hall-effect sensors measure the strength of the local magnetic field, which should increase with proximity to the magnets; however, this is complicated by neighboring magnets in the array. Hence, noisy data were combined using Bayesian algorithms to predict membrane deformation for a deformable fingertip.

Later, *Russell* and *Parkinson* [28.26] developed an impedance tomographic tactile sensor, capable of measure skin deformation over an 8×5 array. This sensor was constructed with of neoprene rubber and filled with distilled water, similar to the more recent BioTac sensor [28.24]. Row and column electrodes were made from copper and conductive rubber for the rigid substrate and neoprene skin, respectively. Like the capacitive tactile sensors described above, this sensor utilized multiplexing electronics to reduce the number of electrical interconnects. Square waveform driver electronics are used to estimate the resistance of a column of water formed between row and column elements providing a signal that is proportional to the current skin height.

Ferrier and *Brocket* [28.23] implemented a tactile sensor which used optical tracking in combination with models of the skin to predict sensor fingertip deformations. The fingertip sensor consists of a tiny CCD camera focused on a 7×7 array of dots marked on the inside of a gel-filled silicone membrane. An algorithm is used to construct a 13×13 grid over the array of dots. This algorithm uses a combination of the position that is sensed by the CCD camera, which provides the location along a line radially outward from the focal point, in combination with a mechanical model used to solve for the radial distance from the camera focal point, based on energy minimization.

A more recent example of a shape sensing array is the commercial Syntouch BioTac that grew out of research on a multimodal fingertip tactile sensor [28.24]. In this design, measuring the changes in resistance between terminals in a water-filled skin provides an estimate of the skin deformation.

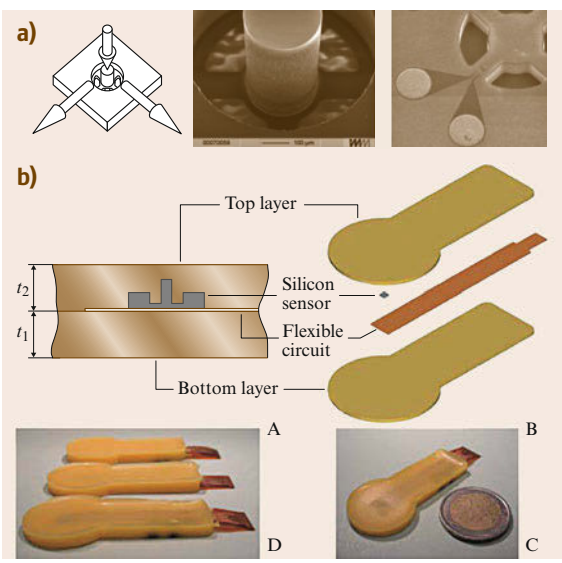


Fig. 28.5 **(a)** Scanning electron microscope (SEM) micrographs of a MEMS 3-axis tactile force sensor. **(b)** MEMS force sensor wirebonded to a flex-circuit and embedded within a silicone rubber skin (after [28.115])

Other Array Tactile Array Sensors

While less common than capacitive or piezoresistive sensors, optical tactile sensors have consistently been attractive for their immunity to electromagnetic interference. The basic approaches include designs that use a small camera to measure skin deformations and arrays of optical emitters and detectors. Optical fibers can also be used to separate the transducer from the contact site [28.42, 119].

An interesting tactile sensor uses vision to track an array of spherical markers embedded in a transparent elastomer to infer the stress state of the skin material

due to applied forces [28.21]. This sensor is currently being commercialized under the tradename GelForce.

Miniaturization of optical components has also made it possible to construct a tactile array of surface-mounted optical devices. In a physically robust design that can be adapted for shear and normal sensing, emitters and detector pairs are covered by a thin translucent layer of silicone rubber followed by an opaque outer layer. As the silicone skin is depressed, there is a variation in the amount of light reflected from the emitter to the detector [28.22, 120].

The concept of reflection off the underside of a compliant skin has also been used with acoustic or ultrasonic sensors. *Shinoda* et al. [28.121] present a sensor that looks at the change in reflected accoustic energy from a resonator chamber near the surface of the skin, with application to friction measurement [28.122]. *Ando* et al. [28.123] present a more sophisticated ultrasound sensor that achieves 6-DOF (degree of freedom) displacement sensing via paired plate elements that utilize 4 ultrasound transducers per plate.

Multimodal Arrays

Human sensing is inherently multi-modal, with combinations of fast- and slow-acting mechanoreceptors as well as specialized thermal and pain receptors. In an analogous approach, a number of researchers have developed multi-modal tactile sensing suites that combine mechanical and thermal sensing [28.24, 49, 52, 124, 125]. As noted earlier, other work has combined static and dynamic sensing [28.24, 29, 32, 46, 92, 93, 126].

28.3 Tactile Information Processing

In discussing the processing of tactile information, we return first to the three main uses depicted in Fig. 28.1. For manipulation, we require, foremost, information about contact locations and forces so that we can grasp objects securely and impart desired forces and motions to them. For exploration, we are concerned with obtaining and integrating information about the object, including the local geometry, hardness, friction, texture, thermal conductivity, etc. For response, we are concerned especially with the detection of events, such as contacts produced by an external agent, and in assessing their types and magnitudes. The uses of information are often coupled. For example, we manipulate objects in order to explore them, and we use the information obtained through object exploration to improve our ability to control forces and motions in manipulation. Recognizing contact events is also important for manipulation and exploration, as it is for response.

28.3.1 Tactile Information Flow: Means and Ends of Tactile Sensing

Figure 28.6 summarizes the general flow of information from each type of sensor reviewed in the previous section through primary sensed quantities to information provided for manipulation, exploration and response. A useful thought exercise is to consider exactly what information we use to perform a task such as turning a pen end-over-end between the fingers. We can easily perform this task with our eyes closed. What information are we using? We need to track the position and orientation of the pen and to monitor the forces that we impose on it to maintain stable manipulation. In other words, we need to know the *configuration* of our grasp, the *locations and movements of contacts* over the surfaces of our fingers, the *magnitudes of grasp forces* and the *contact conditions* with respect to friction limits, etc. The same requirements apply for robots and are provided by the information flow in Fig. 28.6.

At the upper left corner of the figure, joint angles, combined with the forward kinematic model of the manipulator and knowledge of external link geometries, establish the positions and orientations of coordinate frames embedded in the fingertips. This information is needed to integrate local information about object shape, surface normal orientation, etc. so that the overall geometry and pose of the object can be determined.

Actuator effort sensors provide information about the resultant forces, using the Jacobian transpose: $\mathbf{J}^{\mathrm{T}}\boldsymbol{f} = \boldsymbol{\tau}$, where $\boldsymbol{f}$ is an $n \times 1$ vector of external forces and moments taken with respect to a coordinate frame embedded in the appendage. $\mathbf{J}^{\mathrm{T}}$ is the Jacobian transpose, mapping external forces and moments to joint torques, and $\boldsymbol{\tau}$ is an $m \times 1$ vector of joint torques for a serial kinematic chain with m degrees of freedom. We require that the k-th column of $\mathbf{J}^{\mathrm{T}}$ have elements that are relatively large compared to the overall condition number of $\mathbf{J}$ to provide an accurate measurement of the k-th element of $\boldsymbol{f}$. *Eberman* and *Salisbury* [28.127] show that it is possible to measure contact force and location using only joint torque measurements if the manipulator has clean dynamics.

Alternatively, we can use a multi-axis force–torque sensor in the fingers, as shown in Fig. 28.3, or robot wrist to obtain contact forces. This approach has the advantage of providing dynamic force signals with

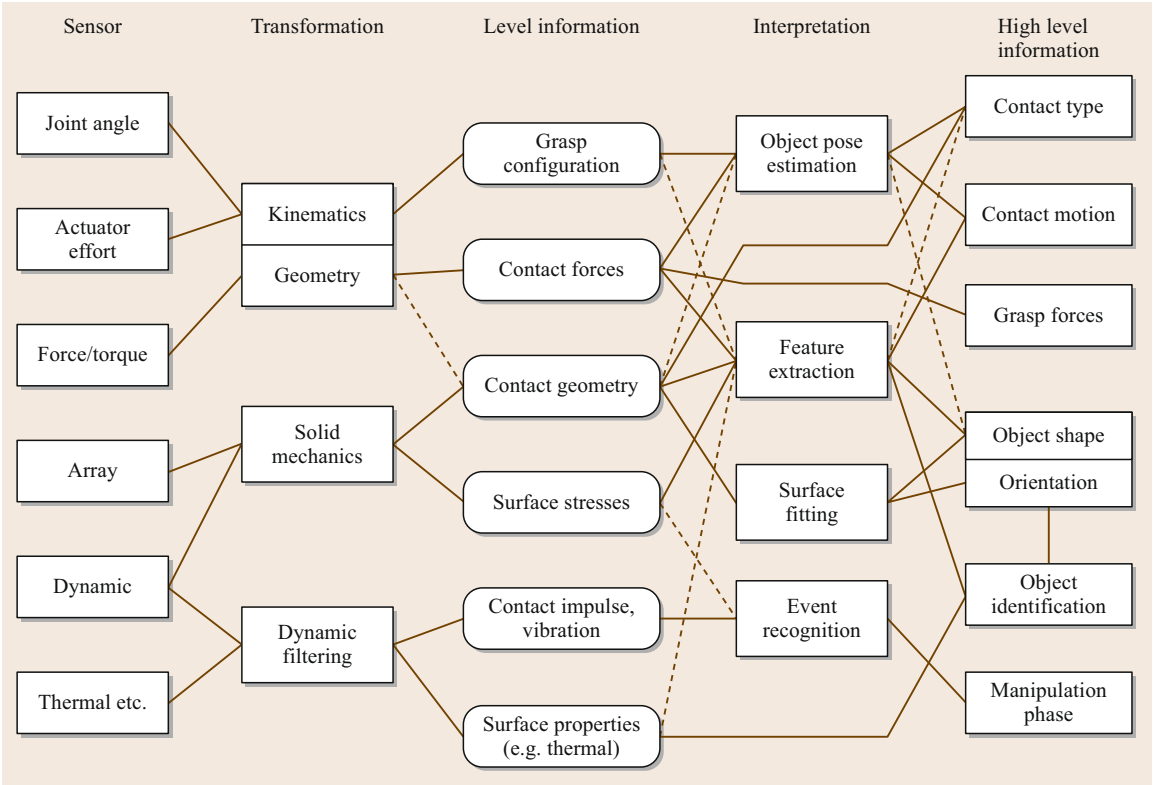


Fig. 28.6 Force and tactile sensor information flow and signal processing

a higher signal/noise ratio because they are not masked by the inertias of the robot arm or fingers and their transmissions. If the geometry of the fingertip is known, one can use *intrinsic tactile sensing* [28.65, 128] to compute the contact location as well as the contact force by examining ratios of resultant forces and torques at the sensor.

When the contacts are small compared to the fingertips (so that a point-contact approximation applies) and the fingertips are convex shapes, the contact location is easily computed. Fig. 28.7 shows a contact force $\boldsymbol{f}$, contacting the fingertip surface at a location $\boldsymbol{r}$. A force–torque sensor such as that in Fig. 28.2 measures the moment, $\boldsymbol{\tau} = \boldsymbol{r} \times \boldsymbol{f}$, with respect to the origin. If we consider the lever arm $\boldsymbol{h}$, perpendicular to the line of action of $\boldsymbol{f}$, then $\boldsymbol{h}/h = \boldsymbol{f}/f \times \boldsymbol{\tau}/\tau$, where $h = \tau/f$ is the magnitude of $\boldsymbol{h}$. We can then write that $\boldsymbol{r} = \boldsymbol{h} - \alpha \boldsymbol{f}$, where α is a constant obtained by solving for the intersection of the line of action and the fingertip surface. For a convex fingertip, there will be two such points, of which only one corresponds to a positive (inward) contact force.

From the contact location one can deduce the local contact normal and contact kinematic type from a small number of force measurements. *Bicchi* presents algorithms for extending these methods to soft fingers [28.128]. *Brock* and *Chiu* [28.66] describe the use of force sensors for the perception of object shape using this approach and for measuring the mass and center of mass of a grasped object.

For precision tasks involving small objects or small forces and motions, cutaneous sensors provide the most sensitive measurements. In general, as task require-

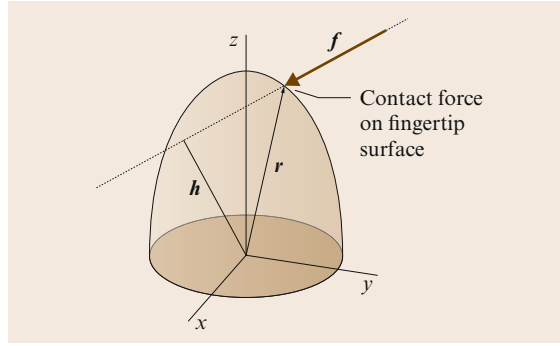


Fig. 28.7 Intrinsic tactile sensing: a contact produces a unique line of action and moment about the origin of a coordinate system in the fingertip. The contact location can be obtained by solving for the intersection of the line of action and the fingertip surface

ments get smaller, the sensor must be located closer to the contact so that the compliance and inertia of the intervening parts of the manipulator do not interfere with the measurement. *Dario* [28.129] suggests that finger tip force sensors are useful for forces of 0.1−10.0 N while array sensors can measure distributed forces of 0.01−1.0 N. *Son* et al. [28.74] find that intrinsic tactile sensing and array sensors can both provide accurate (within 1 mm) estimates of contact location; however, the intrinsic tactile sensing method is inherently sensitive to the accuracy of the force–torque sensor calibration and can produce transient errors due to unmodeled dynamic forces.

Proceeding down the left side of Fig. 28.6 we come to the large category of cutaneous array sensors. The interpretation of information from array sensors depends initially on the transducer type. For arrays of binary contact or proximity sensors, interpretation amounts mainly to establishing the location and general shape of the contact area. Techniques common to binary vision can be used to obtain sub-pixel resolution and to identify contact features. This information, in combination with measurements of the grasp forces from actuator effort or force–torque sensors, is sufficient for basic manipulation tasks [28.69].

28.3.2 Solid Mechanics and Deconvolution

A basic problem associated with tactile array sensors is to reconstruct what is happening at the surface of the skin from a finite set of measurements obtained beneath the surface. Typically we are interested in determining pressure and perhaps shear stress distributions associated with contacts on the skin. In other cases, as when the fingertips consist of a gel or soft foam covered by a thin membrane so that the pressure is nearly constant, the local geometry of the contact is of interest.

In the following example we consider the case of an array of elements located at a depth d below the surface of an elastomeric skin. A contact has resulted in a pressure distribution over the region of interest. We establish a coordinate system with z pointing in the inward normal direction and, for simplicity, we examine a one-dimensional loading case, $p(y)$, in which the pressure distribution is unchanging in the x direction. We further assume that the extent of the skin in the x direction is large compared to the skin thickness so that strains in the x direction are inhibited, leading to a plane strain elasticity problem. We assume that the skin is a homogenous, isotropic material and the strains are small enough that linear elasticity theory can be applied. Of course, none of these assumptions is entirely valid in practical cases; however, the results do agree qualitatively with the measurements obtained with actual robot fingers and tactile arrays. A thorough discussion of the general approach and of the accuracy of the linear elastic models can be found in [28.91, 96, 99, 130, 131].

Figure 28.8 illustrates the case of two line loads, or knife edges, pressed against the surface of the skin (akin to a planar version of the two-point discrimination test for human tactile acuity). The solution for a single line load, or impulse response, was derived by Boussinesq in 1885. For the case of plane strain the principal stresses in the (y, z) plane from a normal unit impulse can be expressed in cartesian coordinates as [28.132]

$$\sigma_z(y,z) = \left(\frac{-2}{\pi z}\right)\frac{1}{[1+(y/z)^2]^2}\,, \tag{28.1}$$

$$\sigma_y(y,z) = \left(\frac{-2}{\pi z}\right)\frac{(y/z)^2}{[1+(y/z)^2]^2}\,, \tag{28.2}$$

$$\sigma_x(y,z) = \nu(\sigma_y+\sigma_z)\,, \tag{28.3}$$

where ν is Poisson's ratio for the material (typically 0.5 for elastomeric materials).

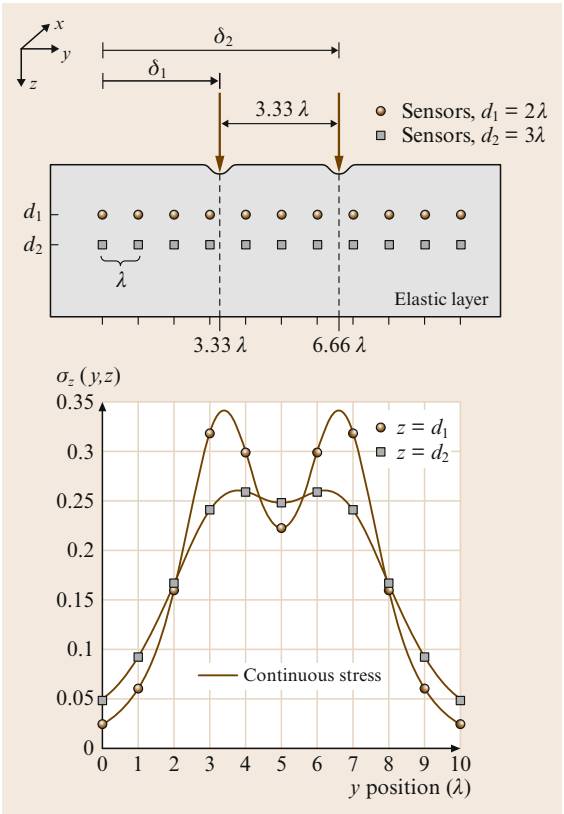


Fig. 28.8 Plane strain stress response for two (unit magnitude) line loads. Note the *blurring* that occurs with greater depth

Avidbots Ex1016
Page 762 of 2154

For two such line loads located at distances δ_1 and δ_2 from the origin, the solution can be obtained by superposition

$$\sigma_z(y,z) = \left(\frac{-2}{\pi z}\right)\left\{\frac{1}{\left[1+\left(\frac{y-\delta_1}{z}\right)^2\right]^2} + \frac{1}{\left[1+\left(\frac{y-\delta_2}{z}\right)^2\right]^2}\right\}, \tag{28.4}$$

$$\sigma_y(y,z) = \left(\frac{-2}{\pi z}\right)\left\{\frac{\left(\frac{y-\delta_1}{z}\right)^2}{\left[1+\left\{\frac{y-\delta_1}{z}\right\}^2\right]^2} + \frac{\left(\frac{y-\delta_2}{z}\right)^2}{\left[1+\left\{\frac{y-\delta_2}{z}\right\}^2\right]^2}\right\}. \tag{28.5}$$

For more general pressure distributions the stresses can be found by convolution of the pressure distribution, $p(y)$, and the impulse response $G_i(y,z)$

$$\sigma_i = \int_{\tau=-\infty}^{\tau=y} [p(\tau)\mathrm{d}\tau] G_i(y-\tau,z)\,. \tag{28.6}$$

Also plotted in Fig. 28.8 are curves corresponding to the vertical stress components, σ_z, at two different depths $d_1 = 2\lambda$ and $d_2 = 3\lambda$, where λ is the sensor spacing. As we go deeper beneath the skin, the stresses become smoothed or blurred, and the ability to distinguish between closely spaced impulses diminishes. However, the blurring of concentrated pressure distributions can also provide an advantage when we have a limited number of sensors because the stresses and strains spread over a larger area and are more likely to affect at least one sensor. The elastic skin also provides a kind of automatic *edge enhancement* because stresses are high at the transitions between loaded and unloaded regions of the skin.

In most cases, for example in the case of capacitive or magnetic sensors, the sensing elements will measure strains or local deformations of the skin material in the vertical direction. In a few cases, such as pieces of piezoelectric film embedded in an elastomeric skin [28.91], the sensors are sufficiently stiff compared to the surrounding material that they can be considered to measure stresses directly.

For the case of elastic plane strain, the strains are related to the stresses by [28.133]

$$\epsilon_y = \frac{1}{E}\left[\sigma_y - \nu(\sigma_x+\sigma_z)\right], \tag{28.7}$$

$$\epsilon_z = \frac{1}{E}\left[\sigma_z - \nu(\sigma_x+\sigma_y)\right], \tag{28.8}$$

where E is the Young's modulus and ν is the Poisson's ratio, which we assume is 0.5 for an elastomeric skin.

Figure 28.9 shows the typical measurements that may be obtained from a row of sensing elements from the two line loads applied in Fig. 28.8. Each bar corresponds to the strain, ϵ_{zi} measured by a corresponding element and computed using (28.8), with stresses obtained from (28.3)–(28.5).

The problem at this point is to produce a best estimate of the surface pressure distribution, $p(y)$, from this finite set of subsurface strain measurements. The problem is a classic example of estimating a signal from a sparse set of remote measurements. One approach to this process is based on deconvolution techniques [28.91, 96, 99]. The measured signal from the sensors $\boldsymbol{\epsilon}_z$ is convolved with the inverse of the impulse strain response $H(y)$ to find an estimate of the surface pressure that produced the signals. The inversion tends to amplify high frequency noise, and the inverse filter bandwidth must be limited according to the spatial density of the sensors and their depth below surface.

Another approach [28.21, 130] is to assume that the surface pressure distribution can be approximated by a finite set of impulses $\boldsymbol{p} = [p_1, p_2 \ldots p_n]^t$. The sensor readings form a vector, $\boldsymbol{\epsilon} = [\epsilon_1, \epsilon_2 \ldots \epsilon_m]^t$, where $m > n$ for the bandwidth limitations discussed above. The strain response can then be written as a matrix

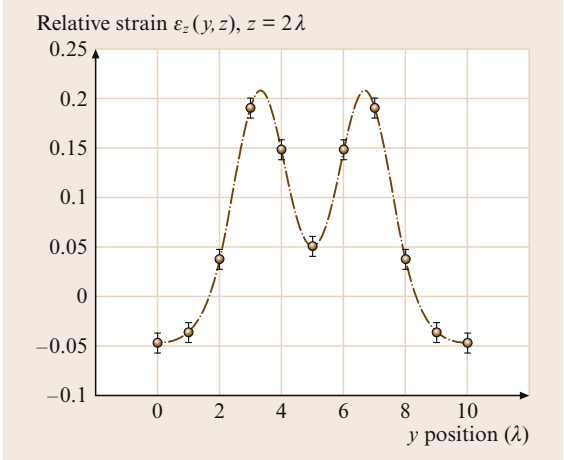


Fig. 28.9 Measured strain with assumed 10% noise

Avidbots Ex1016
Page 763 of 2154

equation

$$\boldsymbol{\epsilon} = \mathbf{H}\boldsymbol{p} \,. \tag{28.9}$$

Each element of $\mathbf{H}$ is computed using (28.8) with σ_z and σ_y computed using equations similar to (28.4) and (28.5) and with σ_x from (28.3). The estimated discrete pressure distribution is then found by taking the pseudoinverse of $\mathbf{H}$

$$\hat{\boldsymbol{p}} = \mathbf{H}^{+}\boldsymbol{\epsilon} \,. \tag{28.10}$$

Using the strain measurements from Fig. 28.9, at a depth $d = 2\lambda$, the estimated pressure distribution using the pseudoinverse method is seen in Fig. 28.10. In this example, because the assumed set of seven impulses matches fortuitously with the actual loading, the reconstruction is fairly accurate despite the assumed noise.

An alternative approach to constructing soft robot finger tips is to enclose a compliant medium such as foam rubber or fluid in a thin elastic membrane [28.25, 27, 117, 134–137]. Some of the tactile array sensors developed for these fingers are able to measure directly the shape of the membrane, so that a physical model is not needed for interpretation of the signal [28.25]. Another sensing scheme uses an array of magnetic sensors at the center of the finger to measure the changes in the magnetic field generated by deformations of the magnet-loaded membrane [28.137]. A statistical algorithm has been developed that can robustly determine the membrane shape from the sensor signals [28.25]. However, a mechanical model is still required to find the pressure distribution across the contact from the shape information provided by all of these sensors.

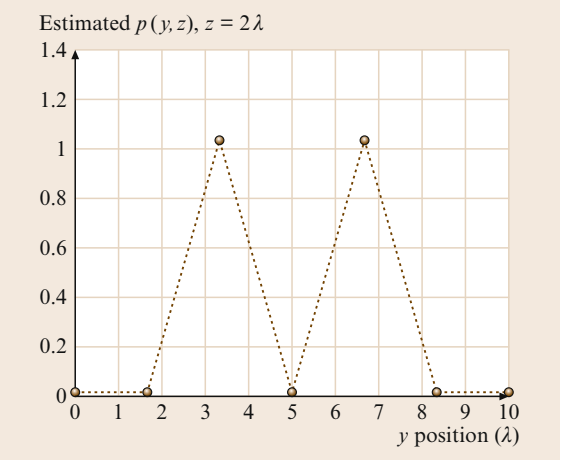


Fig. 28.10 Estimated surface pressure distribution using pseudoinverse method for 11 sensors and 7 assumed impulses

28.3.3 Curvature and Shape Information

Another alternative to measuring subsurface strains or deflections is to measure directly the local curvature at each element of an array of sensors. The curvature information can be applied directly toward identifying contact type and centroid location or integrated to obtain the local shape of the contact [28.28, 138].

Returning to Fig. 28.6, once the local contact shape or geometry has been established, the next steps typically include feature identification (e.g., identifying corners or ridges on an object) and determining the overall shape and pose of the object in the hand.

Often the object shape is at least partially known a priori in which case a variety of surface or data fitting methods can be used. For example, *Fearing* [28.139] developed a method for calculation of the radius of curvature and orientation of a generalized cylinders from tactile array data and [28.140] developed a neural network that performs a similar calculation. Other schemes use contact locations, surface normals and contact forces to determine information about object shape and orientation with respect to the hand [28.141–144].

Allen [28.145] uses several different primitive representations for object shape attributes based on the particular exploratory procedure used to sense the object. Object volume and approximate shape are perceived with enclosure grasping, and the resulting shape is modeled using superquadric surfaces. Similarly, measurement of the lateral extent of object faces leads to a face-edge-vertex model and contour following to a generalized cylinder representation.

The question of what constitutes an appropriate set of features is not well understood, although it clearly depends on the intended application. *Ellis* [28.146] considers appropriate feature sets and methodologies for acquiring the needed data. *Lederman* and *Browse* [28.147] suggest that surface roughness, surface curvature, and oriented edges are used in human haptic perception.

28.3.4 Object and Surface Identification

The most common application of touch information has been in object recognition and classification. In object recognition the goal is to identify one object from a set of known objects using information derived from touch. In classification the goal is to categorize objects according to preselected sensed properties. These systems are usually based on geometric information derived

from tactile array or force sensors, although some approaches use additional information (e.g., compliance, texture, thermal properties and the kinematic pose of the hand) [28.148–154].

28.3.5 Active Sensing Strategies

Because touch provides only local information, movement is an integral part of touch sensing for recognition and exploration. A review of tactile sensing focusing on exploration and manipulation is found in [28.2]. Several researchers have developed strategies for scheduling sensor movements so that each additional observation decreases the number of objects which are consistent with prior observations. This is sometimes described as a *hypothesize and test* approach. Early examples include [28.141, 155–157].

Klatzky et al. [28.149] have suggested that robotics systems can employ the same exploratory procedures used by humans in haptic exploration. These procedures prescribe the finger motions needed for tasks such as tracing object contours, measuring compliance, and determining the lateral extent of object surfaces. Early work on exploratory tactile sensing includes [28.145, 158, 159]. More recent examples include [28.32, 160, 161].

Edge tracking and surface following strategies have received particular attention. Early examples include [28.162–166]; a more recent example is found in [28.167].

28.3.6 Dynamic Sensing and Event Detection

For dynamic tactile sensors used to detect such events as gentle contacts or slippage between the fingertips and an object, the main challenge is to reliably detect the event in question without false positives. The dynamic tactile sensors that produce large signals in response to contact events are also prone to producing large signals in response to vibrations from the robot drive train and to rapid accelerations of the robot hand. Solutions for more robustly detecting contact events include comparing the signals from dynamic tactile sensors at and away from the contact regions and statistical pattern recognition methods to identify the *signature* of true contact events. Early examples include [28.86, 127, 168]. More recent examples have taken advantage of the much greater realtime processing capability that is now available for signal processing and event classification [28.32, 161, 169, 170].

28.3.7 Integration of Thermal and Other Sensors

Sensors such as thermal contact sensors are rarely used in isolation; their signals are generally integrated with those from tactile arrays and other sensors to produce additional information for identifying objects. Examples of integrated thermal and mechanical sensors for surface characterization include [28.24, 52, 171].

28.4 Integration Challenges

A critical problem that we have not yet addressed is the difficulty of connecting to a large and diverse array of tactile sensors. In 1987 *Jacobsen* et al. [28.172] cited the routing of wires as perhaps the most difficult problem in dexterous hand design and, to a large extent, this remains true today. However, some solutions to this problem have been presented in recent years using wireless sensor arrays or digital busses for power and signal connections.

In early work, *Shinoda* and *Oasa* [28.173] embedded tiny wireless sensing elements in an elastic skin that uses an inductive base coil to provide power and signal transmission. *Hakozaki* and *Shinoda* [28.174] embed tactile sensor chips between 2 layers of conductive rubber provide power and serial communication. In other work, [28.18, 20] produce large scale printed arrays with built-in multiplexing and communications.

In a different approach, *Yamada* et al. [28.168] use wireless sensor chips and light transmitted through a transparent elastomer both for power and to communicate to a power-receiver chip. *Ascari* et al. [28.175] propose a communications scheme with all information transmitted along optical fibers.

In recent years, the proliferation of microprocessors that can perform local signal conditioning, multiplexing and digital communications has been particularly helpful.These devices, adapted from touch-screens, are small enough to be located at the fingertips of a hand. One solution is to modify the barometric pressure sensing chips common in cell phones, converting them into pressure sensing elements [28.95, 176]. Another is to use processors adapted for capacitive touch screens [28.94, 103]. Still another involves arrays of optical tactile devices [28.22].

Avidbots Ex1016
Page 765 of 2154

28.5 Conclusions and Future Developments

In comparison to computer vision, tactile sensing always seems to be *a few years away* from widespread adoption. As explained in the introduction to this chapter, the reasons include physical problems (placement and robustness of sensors, wiring challenges) and the diversity of sensor types for detecting forces, pressures, local geometries, vibrations, etc. As we have seen, the transduction and interpretation methods are typically different for each of these tactile quantities. However, there are some basic issues that apply to tactile sensing in general. For example, sensors are generally located within or beneath a compliant skin, which affects the quantities that they sense in comparison to pressures, stresses, thermal gradients or displacements applied to the skin surface.

When choosing tactile sensors for a robot arm or hand, it is effective to begin with a consideration of which tactile quantities are most desired and for what purpose. For example, if the main concern is to obtain accurate measurements of loads or contact forces at sufficient data rates for force servoing, then intrinsic tactile sensing may make the most sense. If manipulating objects with soft contacts and with sliding or rolling, curved array sensors for measuring pressure distributions, or perhaps local skin deflections, may be desirable. If exploring objects to learn about their texture and material composition, dynamic tactile sensors and thermal sensors may be effective.

In an ideal world, one would incorporate all these tactile sensors in a robotic end-effector without regard to cost, signal processing or wiring complexity. Fortunately, the cost and size of transducers suitable for tactile sensing are steadily dropping and the ability to perform localized processing is improving with surface-mounted devices on flexible circuits. In the near future it will be increasingly possible to fabricate dense arrays of transducers in-situ on contoured surfaces, using material deposition and laser machining techniques. In this way, robots may finally start to approach the tactile sensitivity and responsiveness of animals.

Video-References

VIDEO 14 The effect of twice dropping, and then gently placing, a two gram weight on a small capacitive tactile array
available from http://handbookofrobotics.org/view-chapter/28/videodetails/14

VIDEO 15 Capacitive tactile sensing
available from http://handbookofrobotics.org/view-chapter/28/videodetails/15

References

28.1 M.I. Tiwana, S.J. Redmond, N.H. Lovell: A review of tactile sensing technologies with applications in biomedical engineering, Sens. Actuators A Phys. **179**, 17–31 (2012)

28.2 H. Yousef, M. Boukallel, K. Althoefer: Tactile sensing for dexterous in-hand manipulation in robotics – A review, Sens. Actuators A Phys. **167**(2), 171–187 (2011)

28.3 R.S. Dahiya, G. Metta, M. Valle, G. Sandini: Tactile sensing – From humans to humanoids, IEEE Trans. Robotics **26**(1), 1–20 (2010)

28.4 C. Lucarotti, C.M. Oddo, N. Vitiello, M.C. Carrozza: Synthetic and bio-artificial tactile sensing: A review, Sensors **13**(2), 1435–1466 (2013)

28.5 M.H. Lee: Tactile sensing: new directions, new challenges, Int. J. Robotic Res. **19**(7), 636–643 (2000)

28.6 M.H. Lee, H.R. Nicholls: Tactile sensing for mechatronics-a state of the art survey, Mechatronics **9**(1), 1–31 (1999)

28.7 L.D. Harmon: Automated tactile sensing, Int. J. Robotics Res. **1**(2), 3–32 (1982)

28.8 J.-P. Uldry, R.A. Russell: Developing conductive elastomers for applications in robotic tactile sensing, Adv. Robotics **6**(2), 255–271 (1992)

28.9 T.V. Papakostas, J. Lima, M. Lowe: A large area force sensor for smart skin applications, Proc. IEEE Sens., Vol. 2 (2002) pp. 1620–1624

28.10 M. Shimojo, A. Namiki, M. Ishikawa, R. Makino, K. Mabuchi: A tactile sensor sheet using pressure conductive rubber with electrical-wires stitched method, IEEE Sens. J. **4**(5), 589–596 (2004)

28.11 D. De Rossi, A. Della Santa, A. Mazzoldi: Dressware: wearable piezo- and thermoresistive fabrics for ergonomics and rehabilitation, Proc. 19th Annu. Int. Conf. IEEE Eng. Med. Biol. Soc., Vol. 5 (1997) pp. 1880–1883

28.12 A. Tognetti, F. Lorussi, M. Tesconi, D. De Rossi: Strain sensing fabric characterization, Proc. IEEE Sens., Vol. 1 (2004) pp. 527–530

28.13 R.S. Fearing, T.O. Binford: Using a cylindrical tactile sensor for determining curvature, IEEE Trans. Robotics Autom. **7**(6), 806–817 (1991)

Avidbots Ex1016
Page 766 of 2154

28.14 Pressure Profile Systems: http://www.pressureprofile.com/

28.15 H.-K. Lee, S.-I. Chang, E. Yoon: A flexible polymer tactile sensor: Fabrication and modular expandability for large area deployment, J. Microelectromechanical Syst. **15**(6), 1681–1686 (2006)

28.16 T. Hoshi, H. Shinoda: A sensitive skin based on touch-area-evaluating tactile elements, Proc. 14th Symp. Haptic Interfaces Virtual Env. Teleoperator Syst. (2006) pp. 89–94

28.17 P. Maiolino, M. Maggiali, G. Cannata, G. Metta, L. Natale: A flexible and robust large scale capacitive tactile system for robots, IEEE Sens. J. **13**(10), 3910–3917 (2013)

28.18 T. Sekitani, M. Takamiya, Y. Noguchi, S. Nakano, Y. Kato, T. Sakurai, T. Someya: A large-area wireless power-transmission sheet using printed organic transistors and plastic MEMS switches, Nat. Mater. **6**(6), 413–417 (2007)

28.19 R.S. Dahiya, D. Cattin, A. Adami, C. Collini, L. Barboni, M. Valle, L. Lorenzelli, R. Oboe, G. Metta, F. Brunetti: Towards tactile sensing system on chip for robotic applications, IEEE Sens. J. **11**(12), 3216–3226 (2011)

28.20 K. Takei, T. Takahashi, J.C. Ho, H. Ko, A.G. Gillies, P.W. Leu, R.S. Fearing, A. Javey: Nanowire active-matrix circuitry for low-voltage macroscale artificial skin, Nat. Mater. **9**(10), 821–826 (2010)

28.21 K. Kamiyama, H. Kajimoto, N. Kawakami, S. Tachi: Evaluation of a vision-based tactile sensor, Proc. IEEE Int. Conf. Robotics Autom. (ICRA), Vol. 2 (2004) pp. 1542–1547

28.22 M. Quigley, C. Salisbury, A.Y. Ng, J.K. Salisbury: Mechatronic design of an integrated robotic hand, Int. J. Robobotics Res. **33**(5), 706–720 (2014)

28.23 N.J. Ferrier, R.W. Brockett: Reconstructing the shape of a deformable membrane from image data, Int. J. Robotics Res. **19**(9), 795–816 (2000)

28.24 C.H. Lin, T.W. Erickson, J.A. Fishel, N. Wettels, G.E. Loeb: Signal processing and fabrication of a biomimetic tactile sensor array with thermal, force and microvibration modalities, Proc. IEEE Int. Conf. Robotics Biomim. (ROBIO) (2009) pp. 129–134

28.25 W.C. Nowlin: Experimental results on Bayesian algorithms for interpreting compliant tactile sensing data, Proc. IEEE Int. Conf. Robotics Autom. (ICRA), Vol. 1 (1991) pp. 378–383

28.26 R.A. Russell, S. Parkinson: Sensing surface shape by touch, Proc. IEEE Int. Conf. Robotics Autom. (ICRA), Vol. 1 (1993) pp. 423–428

28.27 R.A. Russell: Compliant-skin tactile sensor, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (1987) pp. 1645–1648

28.28 W.R. Provancher, M.R. Cutkosky: Sensing local geometry for dexterous manipulation, Proc. Intl. Symp. Exp. Robotics (2002) pp. 507–516

28.29 P. Dario, R. Lazzarini, R. Magni, S.R. Oh: An integrated miniature fingertip sensor, Proc. 7th Int. Symp. Micro Mach. Hum. Sci. (1996) pp. 91–97

28.30 R.D. Howe, M.R. Cutkosky: Dynamic tactile sensing: perception of fine surface features with stress rate sensing, IEEE Trans. Robotics Autom. **9**(2), 140–151 (1993)

28.31 R.D. Howe, M.R. Cutkosky: Sensing skin acceleration for texture and slip perception, Proc. IEEE Int. Conf. Robotics Autom. (ICRA), Vol. 1 (1989) pp. 145–150

28.32 J.M. Romano, K. Hsiao, G. Niemeyer, S. Chitta, K.J. Kuchenbecker: Human-inspired robotic grasp control with tactile sensing, IEEE Trans. Robotics **27**(6), 1067–1079 (2011)

28.33 J. Lee, S.N. Sponberg, O.Y. Loh, A.G. Lamperski, R.J. Full, N.J. Cowan: Templates and anchors for antenna-based wall following in cockroaches and robots, IEEE Trans. Robotics **24**(1), 130–143 (2008)

28.34 T.J. Prescott, M.J. Pearson, B. Mitchinson, J.C. Sullivan, A. Pipe: Whisking with robots: from rat vibrissae to biomimetic technology for active touch, IEEE Robotics Autom. Mag. **16**(3), 42–50 (2009)

28.35 R.A. Russell: Using tactile whiskers to measure surface contours, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (1992) pp. 1295–1299

28.36 M. Kaneko, N. Kanayama, T. Tsuji: Active antenna for contact sensing, IEEE Trans. Robotics Autom. **14**(2), 278–291 (1998)

28.37 T.N. Clements, C.D. Rahn: Three-dimensional contact imaging with an actuated whisker, IEEE Trans. Robotics **22**(4), 844–848 (2006)

28.38 T.J. Prescott, M.J. Pearson, B. Mitchinson, J.C. Sullivan, A. Pipe: Tactile discrimination using active whisker sensors, IEEE Sens. J. **12**(2), 350–362 (2012)

28.39 J.M. Vranish, R.L. McConnell, S. Mahalingam: Capaciflector collision avoidance sensors for robots, Comput. Electr. Eng. **17**(3), 173–179 (1991)

28.40 E. Cheung, V. Lumelsky: A sensitive skin system for motion control of robot arm manipulators, Robotics Auton. Syst. **10**(1), 9–32 (1992)

28.41 D. Um, V. Lumelsky: Fault tolerance via component redundancy for a modularized sensitive skin, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (1999) pp. 722–727

28.42 S. Walker, K. Loewke, M. Fischer, C. Liu, J.K. Salisbury: An optical fiber proximity sensor for haptic exploration, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (2007) pp. 473–478

28.43 P. Wei, L. Zhizeng: A design of miniature strong anti-jamming proximity sensor, Proc. Int. Conf. Comp. Sci. Electron. Eng. (ICCSEE) (2012) pp. 327–331

28.44 E. Guglielmelli, V. Genovese, P. Dario, G. Morana: Avoiding obstacles by using a proximity US/IR sensitive skin, IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS) (1993) pp. 2207–2214

28.45 D. Wegerif, D. Rosinski: Sensor based whole arm obstacle avoidance for kinematically redundant robots, Proc. SPIE – Int. Soc. Opt. Eng. **1828**, 417–426 (1992)

28.46 G. Buttazzo, P. Dario, R. Bajcsy: Finger based explorations, Proc. SPIE 0726, Intell. Robots Comput. Vis. V, ed. by D.P. Casadent (1986) pp. 338–345

Avidbots Ex1016
Page 767 of 2154

28.47 D. Siegel, I. Garabieta, J. Hollerbach: An integrated tactile and thermal sensor, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (1986) pp. 1286–1291

28.48 R.A. Russell: A thermal sensor array to provide tactile feedback for robots, Int. J. Robotics Res. **5**(3), 35–39 (1985)

28.49 F. Castelli: An integrated tactile-thermal robot sensor with capacitive tactile array, IEEE Trans. Ind. Appl. **38**(1), 85–90 (2002)

28.50 D.G. Caldwell, C. Gosney: Enhanced tactile feedback (Tele-taction) using a multi-functional sensory system, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) **1**, 955–960 (1993)

28.51 G.J. Monkman, P.M. Taylor: Thermal tactile sensing, IEEE Trans. Robotics Autom. **9**(3), 313–318 (1993)

28.52 J. Engel, J. Chen, X. Wang, Z. Fan, C. Liu, D. Jones: Technology development of integrated multimodal and flexible tactile skin for robotics applications, Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS), Vol. 3 (2003) pp. 2359–2364

28.53 P. Bergveld: Development and application of chemical sensors in liquids. In: *Sensors and Sensory Systems for Advanced Robots*, NATO ASI Series, Vol. 43, ed. by P. Dario (Springer, Berlin, Heidelberg 1988) pp. 397–414

28.54 T. Nakamoto, A. Fukuda, T. Moriizumi: Perfume and flavor identification by odor sensing system using quartz-resonator sensor array and neural-network pattern recognition, Proc. 6th Int. Conf. Solid-State Sens. Actuators (TRANSDUCERS '91) (1991)

28.55 R.A. Russell: Survey of robotic applications for odor-sensing technology, Int. J. Robotics Res. **20**(2), 144–162 (2001)

28.56 A.J. Lilienthal, A. Loutfi, T. Duckett: Airborne chemical sensing with mobile robots, Sensors **6**(11), 1616–1678 (2006)

28.57 B.A. Auld, A.J. Bahr: A novel multifunction robot sensor, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (1986) pp. 1791–1797

28.58 H. Clergeot, D. Placko, J.M. Detriche: Electrical proximity sensors. In: *Sensors and Sensory Systems for Advanced Robots*, NATO ASI Series, Vol. 43, ed. by P. Dario (Springer, Berlin, Heidelberg 1988) pp. 295–308

28.59 M. Kaneko, K. Tanie: Contact point detection for grasping of an unknown object using self-posture changeability (SPC), IEEE Trans. Robotics Autom., Vol. 10 (1994) pp. 355–367

28.60 A.M. Dollor, L.P. Jentoft, J.H. Cao, R.D. Howe: Contact sensing and grasping performance of compliant hands, Auton. Robots **28**(1), 65–75 (2010)

28.61 J.K. Salisbury: Appendix to kinematic and force analysis of articulated hands. In: *Robot Hands and the Mechanics of manipulation*, ed. by M.T. Mason, J.K. Salisbury (MIT Press, Cambridge 1985)

28.62 G. Palli, C. Melchiorri, G. Vassura, U. Scarcia, L. Moriello, G. Berselli, A. Cavallo, G. De Maria, C. Natale, S. Pirozzi, C. May, F. Ficuciello, B. Siciliano: The DEXMART hand: Mechatronic design and experimental evaluation of synergy-based control for human-like grasping, Int. J. Robotics Res. **33**(5), 799–824 (2014)

28.63 A. Pugh (Ed.): *Robot Sensors, Volume 2: Tactile and Non-Vision* (IFS Publ./Springer, New York 1986)

28.64 J.G. Webster: *Tactile Sensors for Robotics and Medicine* (Wiley, New York 1988)

28.65 J.K. Salisbury: Interpretation of contact geometries from force measurements. In: *Robotics Res. First Int. Symp*, ed. by M. Brady, R.P. Paul (MIT Press, Cambridge 1984)

28.66 D. Brock, S. Chiu: Environment perception of an articulated robot hand using contact sensors, ASME Winter Annu. Meet. Robotics Manuf. Automa., Vol. 15 (1985) pp. 89–96

28.67 J. Butterfass, M. Grebenstein, H. Liu, G. Hirzinger: DLR-Hand II: next generation of a dextrous robot hand, Proc. IEEE Int. Conf. Robotics Autom. (ICRA), Vol. 1 (2001) pp. 109–114

28.68 F.W. Sinden, R.A. Boie: A planar capacitive force sensor with six degrees of freedom, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (1986) pp. 1806–1813

28.69 B.B. Edin, L. Beccai, L. Ascari, S. Roccella, J.J. Cabibihan, M.C. Carrozza: A bio-inspired approach for the design and characterization of a tactile sensory system for a cybernetic prosthetic hand, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (2006) pp. 1354–1358

28.70 Y.-L. Park, S.C. Ryu, R.J. Black, K.K. Chau, B. Moslehi, M.R. Cutkosky: Exoskeletal force-sensing end-effectors with embedded optical fiber-bragg-grating sensors, IEEE Trans. Robotics **25**(6), 1319–1331 (2009)

28.71 A. Bicchi: A criterion for optimal design of multiaxis force sensors, Robotics Auton. Syst. **10**(4), 269–286 (1992)

28.72 M. Uchiyama, E. Bayo, E. Palma-Villalon: A mathematical approach to the optimal structural design of a robot force sensor, Proc. USA-Japan Symp. Flexible Automation (1998) pp. 539–546

28.73 A. Bicchi, J.K. Salisbury, P. Dario: Augmentation of grasp robustness using intrinsic tactile sensing, Proc. IEEE Int. Conf. Robotics Autom. (ICRA), Vol. 1 (1989) pp. 302–307

28.74 J.S. Son, M.R. Cutkosky, R.D. Howe: Comparison of contact sensor localization abilities during manipulation, Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS), Vol. 2 (1995) pp. 96–103

28.75 R.S. Johansson, J.R. Flanagan: Coding and use of tactile signals from the fingertips in object manipulation tasks, Nat. Rev. Neurosci. **10**(5), 345–359 (2009)

28.76 M. Ueda: Tactile sensors for an industrial robot to detect a slip, Proc. 2nd Int. Symp. Ind. Robots (1972) pp. 63–70

28.77 R. Matsuda: Slip sensor of industrial robot and its application, Electric. Eng. Jap. **96**(5), 129–136 (1976)

28.78 J. Rebman, J.-E. Kallhammer: A Search for Precursors of Slip in Robotic Grasp, Intelligent Robots

Avidbots Ex1016
Page 768 of 2154

and Computer Vision: Fifth in a Series, Cambridge, ed. by E. Casaent (1986) pp. 329–337
28.79 P. Dario, D. De Rossi: Tactile sensors and the gripping challenge, IEEE Spectrum **22**(8), 46–52 (1985)
28.80 R.W. Patterson, G.E. Nevill: The induced vibration touch sensor – A new dynamic touch sensing concept, Robotica **4**(01), 27–31 (1986)
28.81 M.R. Cutkosky, J. Ulmen: Dynamic Tactile Sensing. In: *The Human Hand as an Inspiration for Robot Hand Development*, Springer Tracts in Advanced Robotics 95, ed. by R. Balasubramanian, V.J. Santos (Springer, Cham 2014) pp. 389–403
28.82 D. Dornfeld, C. Handy: Slip detection using acoustic emission signal analysis, Proc. IEEE Int. Conf. Robotics Autom. (ICRA), Vol. 3 (1987) pp. 1868–1875
28.83 X.A. Wu, N. Burkhard, B. Heyneman, R. Valen, M.R. Cutkosky: Contact event detection for robotic oil drilling, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (2014) pp. 2255–2261
28.84 S. Omata: Real time robotic tactile sensor system for the determination of the physical properties of biomaterials, Sens. Actuators A Phys. **112**(2/3), 278–285 (2004)
28.85 S.B. Backus, A.M. Dollar: Robust resonant frequency-based contact detection with applications in robotic reaching and grasping, IEEE/ASME Trans. Mechatron. **19**(5), 1552–1561 (2014)
28.86 M.R. Tremblay, M.R. Cutkosky: Estimating friction using incipient slip sensing during a manipulation task, Proc. IEEE Int. Conf. Robotics Autom. (ICRA), Vol. 1 (1993) pp. 429–434
28.87 E.G.M. Holweg, H. Hoeve, W. Jongkind, L. Marconi, C. Melchiorri, C. Bonivento: Slip detection by tactile sensors: Algorithms and experimental results, Proc. IEEE Int. Conf. Robotics Autom. (ICRA), Vol. 4 (1996) pp. 3234–3239
28.88 I. Fujimoto, Y. Yamada, T. Maeno, T. Morizono, Y. Umetani: Identification of incipient slip phenomena based on the circuit output signals of PVDF film strips embedded in artificial finger ridges, Trans. Soc. Instrum. Control Eng. **40**(6), 648–655 (2004)
28.89 B. Choi, H.R. Choi, S. Kang: Development of tactile sensor for detecting contact force and slip, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (2005) pp. 2638–2643
28.90 P.A. Schmidt, E. Maël, R.P. Würtz: A sensor for dynamic tactile information with applications in human–robot interaction and object exploration, Robotics Auton. Syst. **54**(12), 1005–1014 (2006)
28.91 R.D. Howe: Tactile sensing and control of robotic manipulation, Adv. Robotics **8**(3), 245–261 (1993)
28.92 C. Melchiorri: Slip detection and control using tactile and force sensors, IEEE/ASME Trans. Mechatron. **5**(3), 235–243 (2000)
28.93 C.M. Oddo, L. Beccai, G.G. Muscolo, M.C. Carrozza: A biomimetic MEMS-based tactile sensor array with fingerprints integrated in a robotic fingertip for artificial roughness encoding, Proc. IEEE Int. Conf. Robotics Biomim. (2009) pp. 894–900
28.94 A. Schmitz, M. Maggiali, L. Natale, B. Bonino, G. Metta: A tactile sensor for the fingertips of the humanoid robot iCub, Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (2010) pp. 2212–2217
28.95 L.P. Jentoft, Y. Tenzer, D. Vogt, R.J. Wood, R.D. Howe: Flexible, stretchable tactile arrays from MEMS barometers, Proc. 16th Int. Conf. Adv. Robotics (2013) pp. 1–6
28.96 R.S. Fearing, J.M. Hollerbach: Basic solid mechanics for tactile sensing, Int. J. Robotics Res. **4**(3), 40–54 (1985)
28.97 W. Griffin, W.M. Provancher, M.R. Cutkosky: Feedback strategies for telemanipulation with shared control of object handling forces, Presence Teleoperations Virtual Environ. **14**(6), 720–731 (2005)
28.98 H. Maekawa, K. Tanie, K. Komoriya, M. Kaneko, C. Horiguchi, T. Sugawara: Development of a finger-shaped tactile sensor and its evaluation by active touch, Proc. IEEE Int. Conf. Robotics Autom. (ICRA), Vol. 2 (1992) pp. 1327–1334
28.99 R.S. Fearing: Tactile sensing mechanisms, Int. J. Robotics Res. **9**(3), 3–23 (1987)
28.100 G. Cannata, M. Maggiali, G. Metta, G. Sandini: An embedded artificial skin for humanoid robots, Proc. IEEE Int. Conf. Muiltisens. Fusion Integr. Intell. Syst. (2008) pp. 434–438
28.101 M.-Y. Cheng, X.-H. Huang, C.-W. Ma, Y.-J. Yang: A flexible capacitive tactile sensing array with floating electrodes, J. Micromechanics Microengineering **19**(11), 115001 (2009)
28.102 Y. Hasegawa, M. Shikida, D. Ogura, Y. Suzuki, K. Sato: Fabrication of a wearable fabric tactile sensor produced by artificial hollow fiber, J. Micromechanics Microengineering **18**(8), 085014 (2008)
28.103 D. McConnell Aukes, M.R. Cutkosky, S. Kim, J. Ulmen, P. Garcia, H. Stuart, A. Edsinger: Design and testing of a selectively compliant underactuated hand, Int. J. Robotics Res. **33**(5), 721–735 (2014)
28.104 O. Kerpa, K. Weiss, H. Worn: Development of a flexible tactile sensor system for a humanoid robot, Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS) (2003) pp. 1–6
28.105 D. Bloor, A. Graham, E.J. Williams, P.J. Laughlin, D. Lussey: Metal–polymer composite with nanostructured filler particles and amplified physical properties, Appl. Phys. Lett. **88**(10), 102103 (2006)
28.106 Peratech: Peratech QTC, http://www.peratech.com/standard-products/ (2014)
28.107 T. Someya: Integration of organic field-effect transistors and rubbery pressure sensors for artificial skin applications, Proc. IEEE Int. Electron. Dev. Meet. (2003) pp. 8–14
28.108 H. Alirezaei, A. Nagakubo, Y. Kuniyoshi: A tactile distribution sensor which enables stable measurement under high and dynamic stretch, Proc. IEEE Symp. 3D User Interfaces (2009) pp. 87–93
28.109 Y.-L. Park, B.-R. Chen, R.J. Wood: Design and fabrication of soft artificial skin using embedded microchannels and liquid conductors, IEEE Sens. J. **12**(8), 2711–2718 (2012)
28.110 R. Kageyama, S. Kagami, M. Inaba, H. Inoue: Development of soft and distributed tactile sensors

Avidbots Ex1016
Page 769 of 2154

and the application to a humanoid robot, Proc. IEEE Int. Conf. Syst. Man Cybern., Vol. 2 (1999) pp. 981–986
28.111 B.J. Kane, M.R. Cutkosky, G.T.A. Kovacs: A traction stress sensor array for use in high-resolution robotic tactile imaging, J. Microelectromechanical Syst. **9**(4), 425–434 (2000)
28.112 H. Takao, K. Sawada, M. Ishida: Monolithic silicon smart tactile image sensor with integrated strain sensor array on pneumatically swollen single-diaphragm structure, IEEE Trans. Electron. Dev. **53**(5), 1250–1259 (2006)
28.113 K. Noda, I. Shimoyama: A Shear stress sensing for robot hands -Orthogonal arrayed piezoresistive cantilevers standing in elastic material-, Proc. 14th Symp. Haptic Interfaces Virtual Env. Teleoperator Syst. (2006) pp. 63–66
28.114 M.-Y. Cheng, C.-L. Lin, Y.-J. Yang: Tactile and shear stress sensing array using capacitive mechanisms with floating electrodes, 2010 IEEE 23rd Int. Conf. Micro Electro Mech. Syst. (2010) pp. 228–231
28.115 P. Valdastri, S. Roccella, L. Beccai, E. Cattin, A. Menciassi, M.C. Carrozza, P. Dario: Characterization of a novel hybrid silicon three-axial force sensor, Sens. Actuators A **123/124**, 249–257 (2005)
28.116 S.C.B. Mannsfeld, B.C.-K. Tee, R.M. Stoltenberg, C.V.H.H. Chen, S. Barman, B.V.O. Muir, A.N. Sokolov, C. Reese, Z. Bao: Highly sensitive flexible pressure sensors with microstructured rubber dielectric layers, Nat. Mater. **9**(10), 859–864 (2010)
28.117 R. Brockett: Robotic hands with rheological surfaces, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (1985) pp. 942–946
28.118 K.B. Shimoga, A.A. Goldenberg: Soft robotic fingertips. I. A comparison of construction materials, Int. J. Rob, Res. **15**(4), 320–350 (1996)
28.119 A. Mazid, R. Russell: A robotic opto-tactile sensor for assessing object surface texture, IEEE Conf. Robotics Autom. Mechatronics (2006) pp. 1–5
28.120 L.S. Lincoln, S.J.M. Bamberg, E. Parsons, C. Salisbury, J. Wheeler: An elastomeric insole for 3-axis ground reaction force measurement, Proc. IEEE RAS/EMBS Int. Conf. Biomedical Robotics Biomech. (2012) pp. 1512–1517
28.121 H. Shinoda, K. Matsumoto, S. Ando: Acoustic resonant tensor cell for tactile sensing, Proc. IEEE Int. Conf. Robotics Autom. (ICRA), Vol. 4 (1997) pp. 3087–3092
28.122 H. Shinoda, S. Sasaki, K. Nakamura: Instantaneous evaluation of friction based on ARTC tactile sensor, Proc. IEEE Int. Conf. Robotics Autom. (ICRA), Vol. 3 (2000) pp. 2173–2178
28.123 S. Ando, H. Shinoda, A. Yonenaga, J. Terao: Ultrasonic six-axis deformation sensing, IEEE Trans. Ultrason. Ferroelectr. Freq. Control. **48**(4), 1031–1045 (2001)
28.124 P. Dario, D. De Rossi, C. Domenici, R. Francesconi: Ferroelectric polymer tactile sensors with anthropomorphic features, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (1984) pp. 332–340
28.125 D.M. Siegel: *Contact sensors for dextrous robotic hands*, MIT Artificial Intelligence Laboratory Tech. Rep., no. 900 (MIT Press, Cambridge 1986)
28.126 J.S. Son, E.A. Monteverde, R.D. Howe: A tactile sensor for localizing transient events in manipulation, Proc. IEEE Int. Conf. Robotics Autom. (ICRA), Vol. 1 (1994) pp. 471–476
28.127 B.S. Eberman, J.K. Salisbury: Determination of Manipulator Contact Information from Joint Torque Measurements. In: *Experimental Robotics I, The First International Symposium*, ed. by V. Hayward, O. Khatib (Springer, Montreal 1990)
28.128 A. Bicchi: Intrinsic contact sensing for soft fingers, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (1990) pp. 968–973
28.129 P. Dario: Tactile sensing for robots: Present and future. In: *The Robotics Review 1*, ed. by O. Khatib, J. Craig, T. Lozano-Perez (MIT Press, Cambridge 1989) pp. 133–146
28.130 J.R. Phillips, K.O. Johnson: Tactile spatial resolution III: A continuum mechanics model of skin predicting mechanoreceptor responses to bars, edges and gratings, J. Neurophysiol. **46**(6), 1204–1225 (1981)
28.131 T. Speeter: A tactile sensing system for robotic manipulation, Int. J. Robotics Res. **9**(6), 25–36 (1990)
28.132 K.L. Johnson: *Contact Mechanics* (Cambridge Univ. Press, Cambridge 1985)
28.133 S. Timoshenko, J.N.N. Goodier: *Theory of Elasticity* (McGraw-Hill, New York 1951)
28.134 G. Kenaly, M. Cutkosky: Electrorheological fluid-based fingers with tactile sensing, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (1989) pp. 132–136
28.135 R.D. Howe: Dynamic Tactile Sensing, Ph.D. Thesis (Stanford University, Stanford 1990)
28.136 R.M. Voyles, B.L. Stavnheim, B. Yap: Practical electrorheological fluid-based fingers for robotic applications, IASTED Int. Symp. Robotics Manuf. (1989)
28.137 J.J. Clark: A magnetic field based compliance matching sensor for high resolution, high compliance tactile sensing, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (1989) pp. 772–777
28.138 T.H. Speeter: Analysis and Control of Robotic Manipulation, Ph.D. Thesis (Case Western Reserve University, Cleveland 1987)
28.139 R. Fearing: Tactile sensing for shape interpretation. In: *Dextrous Robot Hands*, ed. by S.T. Venkataraman, T. Iberall (Springer, Berlin, Heidelberg 1990) pp. 209–238
28.140 A.J. Worth, R.R. Spencer: A neural network for tactile sensing: The hertzian contact problem, Proc. Int. Jt. Conf. Neural Netw. (1989) pp. 267–274
28.141 W.E.L. Grimson, T. Lozano-Perez: Model-based recognition and localization from sparse range or tactile data, Int. J. Robotics Res. **3**(3), 3–35 (1984)
28.142 P.C. Gaston, T. Lozano-Perez: Tactile recognition and localization using object models: The case of polyhedra on a plane, Proc. IEEE Trans. Pattern Anal. Mach. Intell. (1984) pp. 257–266

Avidbots Ex1016
Page 770 of 2154

28.143 J.L. Schneiter: An objective sensing strategy for object recognition and localization, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (1986) pp. 1262–1267

28.144 R. Cole, C. Yap: Shape from probing, J. Algorithm. **8**(1), 19–38 (1987)

28.145 P.K. Allen: Mapping haptic exploratory procedures to multiple shape representations, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (1990) pp. 1679–1684

28.146 R.E. Ellis: Extraction of tactile features by passive and active sensing, Proc. SPIE 0521 (1985) p. 289

28.147 S.J. Lederman, R. Browse: The physiology and psychophysics of touch. In: *Sensors and Sensory Systems for Advanced Robotics*, ed. by P. Dario (Springer, Berlin, Heidelberg 1986) pp. 71–91

28.148 H. Ozaki, S. Waku, A. Mohri, M. Takata: Pattern recognition of a grasped object by unit-vector distribution, IEEE Trans. Syst. Man Cybern. **12**(3), 315–324 (1982)

28.149 R.L. Klatzky, R. Bajcsy, S.J. Lederman: Object exploration in one and two fingered robots, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (1987) pp. 1806–1809

28.150 D. Siegel: Finding the pose of an object in the hand, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (1991) pp. 406–411

28.151 D. Taddeucci, C. Laschi, R. Lazzarini, R. Magni, P. Dario, A. Starita: An approach to integrated tactile perception, Proc. IEEE Int. Conf. Robotics Autom. (ICRA), Vol. 4 (1997) pp. 3100–3105

28.152 A. Schneider, J. Sturm, C. Stachniss, M. Reisert, H. Burkhardt, W. Burgard: Object identification with tactile sensors using bag-of-features, IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS) (2009) pp. 243–248

28.153 N. Gorges, S.E. Navarro, D. Göger, H. Wörn: Haptic object recognition using passive joints and haptic key features, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (2010) pp. 2349–2355

28.154 Y. Bekiroglu, J. Laaksonen, J.A. Jorgensen, V. Kyrki, D. Kragic: Assessing grasp stability based on learning and haptic data, IEEE Trans. Robotics **27**(3), 616–629 (2011)

28.155 V.S. Gurfinkel: Tactile sensitizing of manipulators, Eng. Cybern. **12**(6), 47–56 (1974)

28.156 R. Ellis: Acquiring tactile data for the recognition of planar objects, Proc. IEEE Int. Conf. Robotics Autom. (ICRA), Vol. 4 (1987) pp. 1799–1805

28.157 A. Cameron: Optimal tactile sensor placement, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (1989) pp. 308–313

28.158 P. Dario: Sensing body structures by an advanced robot system, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (1988) pp. 1758–1763

28.159 S.A.A. Stansfield: Robotic grasping of unknown objects: A knowledge-based approach, Int. J. Robotics Res. **10**(4), 314–326 (1991)

28.160 A. Petrovskaya, O. Khatib: Global localization of objects via touch, IEEE Trans. Robotics **27**(3), 569–585 (2011)

28.161 N.F. Lepora, U. Martinez-Hernandez, H. Barron-Gonzalez, M. Evans, G. Metta, T.J. Prescott: Embodied hyperacuity from Bayesian perception: Shape and position discrimination with an iCub fingertip sensor, IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS) (2012) pp. 4638–4643

28.162 C. Muthukrishnan, D. Smith, D. Meyers, J. Rebman, A. Koivo: Edge detection in tactile images, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (1987) pp. 1500–1505

28.163 A.D. Berger, P.K. Khosla: Using tactile data for real-time feedback, Int. J. Robotics Res. **10**(2), 88–102 (1991)

28.164 K. Pribadi, J.S. Bay, H. Hemami: Exploration and dynamic shape estimation by a robotic probe, IEEE Trans. Syst. Man Cybern. **19**(4), 840–846 (1989)

28.165 H. Zhang, N.N. Chen: Control of contact via tactile sensing, IEEE Trans. Robotics Autom. **16**(5), 482–495 (2000)

28.166 A.M. Okamura, M.R. Cutkosky: Feature detection for haptic exploration with robotic fingers, Int. J. Robotics Res. **20**(12), 925–938 (2001)

28.167 K. Suwanratchatamanee, M. Matsumoto, S. Hashimoto: Robotic tactile sensor system and applications, IEEE Trans. Ind. Electron. **57**(3), 1074–1087 (2010)

28.168 K. Yamada, K. Goto, Y. Nakajima, N. Koshida, H. Shinoda: A sensor skin using wire-free tactile sensing elements based on optical connection, Proc. 41st SICE Annu. Conf., Vol. 1 (2002) pp. 131–134

28.169 M. Schoepfer, C. Schuermann, M. Pardowitz, H. Ritter: Using a piezo-resistive tactile sensor for detection of incipient slippage, Proc. ISR/ROBOTIK 41st Int. Symp. Robotics (2010) pp. 14–20

28.170 B. Heyneman, M.R. Cutkosky: Slip interface classification through tactile signal coherence, IEEE/RSJ IEEE Int. Conf. Intell. Robots Syst. (IROS) (2013) pp. 801–808

28.171 P. Dario, P. Ferrante, G. Giacalone, L. Livaldi, B. Allotta, G. Buttazzo, A.M. Sabatini: Planning and executing tactile exploratory procedures, IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS), Vol. 3 (1992) pp. 1896–1903

28.172 S.C. Jacobsen, J.E. Wood, D.F. Knutti, K.B. Biggers: The Utah/MIT dextrous hand: Work in progress. In: *First International Conference on Robotics Research*, ed. by M. Brady, R.P. Paul (MIT Press, Cambridge 1984) pp. 601–653

28.173 H. Shinoda, H. Oasa: Passive wireless sensing element for sensitive skin, IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS), Vol. 2 (2000) pp. 1516–1521

28.174 M. Hakozaki, H. Shinoda: Digital tactile sensing elements communicating through conductive skin layers, Proc. IEEE Int. Conf. Robotics Autom. (ICRA'02), Vol. 4 (2002) pp. 3813–3817

28.175 L. Ascari, P. Corradi, L. Beccai, C. Laschi: A miniaturized and flexible optoelectronic sensing system for a tactile skin, Int. J. Micromechanics Microengineering **17**, 2288–2298 (2007)

28.176 M. Zillich, W. Feiten: A versatile tactile sensor system for covering large and curved surface areas, IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS) (2012) pp. 20–24

Avidbots Ex1016
Page 771 of 2154

Multimedia Contents



29. Inertial Sensing, GPS and Odometry

Gregory Dudek, Michael Jenkin

This chapter examines how certain properties of the world can be exploited in order for a robot or other device to develop a model of its own motion or pose (position and orientation) relative to an external frame of reference. Although this is a critical problem for many autonomous robotic systems, the problem of establishing and maintaining an orientation or position estimate of a mobile agent has a long history in terrestrial navigation.

29.1 **Odometry** 737
29.2 **Gyroscopic Systems** 739
29.2.1 Mechanical Systems 739
29.2.2 Optical Systems 740
29.2.3 MEMS 741
29.3 **Accelerometers** 742
29.3.1 Mechanical Accelerometer 742
29.3.2 Piezoelectric Accelerometer 743
29.4 **IMU Packages** 743
29.4.1 Performance 744
29.5 **Satellite-Based Positioning (GPS and GNSS)** 744
29.5.1 Overview 745
29.5.2 Performance Factors 746
29.5.3 Enhanced GPS 748
29.5.4 Nationwide Differential GPS System (NDGPS) 748
29.5.5 GPS Receivers and Communications 749
29.6 **GPS-IMU Integration** 749
29.7 **Further Reading** 750
29.7.1 Odometry 750
29.7.2 Gyroscopic Systems 750
29.7.3 Accelerometers 750
29.7.4 GPS 750
29.8 **Currently Available Hardware** 750
29.8.1 Gyroscopic Systems 750
29.8.2 Accelerometers 750
29.8.3 IMU Packages 751
29.8.4 GPS Components 751
References 751

29.1 Odometry

The word *odometry* is a contraction of the Greek words *hodos* meaning *travel* or *journey*, and *metron* meaning measure. Given its importance to a wide variety of applications from civil engineering to military conquest, the basic concepts that underly odometry have been studied for over 2000 years. Perhaps the earliest reference to odometry appears in his *Ten Books on Architecture* where *Vitruvius* describes [29.1]

> *a useful invention of the greatest ingenuity, transmitted by our predecessors, which enables us, while sitting in a carriage on the road or sailing by sea, to know how many miles of a journey we have accomplished.*

In the context of autonomous vehicles, odometry usually refers to the use of data from the actuators (wheels, treads, etc.) to estimate the overall motion of the vehicle. The basic concept [29.2] is to develop a mathematical model of how commanded motions of the vehicles wheels/joints/etc. induce motion of the vehicle itself, and then to integrate these commanded motions over time in order to develop a model of the pose of the vehicle as a function of time. The use of odometry infor-

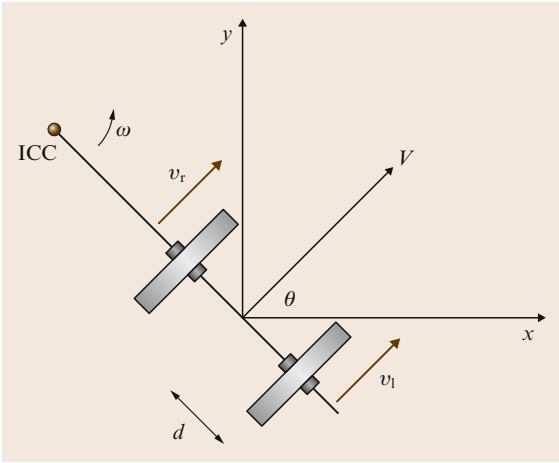


Fig. 29.1 Differential drive kinematics

mation to estimate the pose of the vehicle as a function of time is known as *Dead Reckoning* or *Deductive Reckoning* and finds wide application in navigation at sea [29.3].

The details of odometry estimation varies with vehicle design. In the context of mobile robots perhaps the simplest vehicle for odometry estimation is the differential drive vehicle (Fig. 29.1). A differential drive vehicle has two driveable wheels which are independently controllable and which are mounted along a common axis. Assuming that the location of the wheels are fixed on the vehicle, then for the wheels to remain in constant contact with the ground, the two wheels must describe arcs on the plane such that the vehicle rotates around a point (known as the ICC – instantaneous center of curvature) that lies on the wheels' common axis (Fig. 29.1). If the ground contact speeds of the left and right wheels are v_l and v_r respectively, and the wheels are separated by a distance $2d$, then

$$\omega(R+d) = v_r \,,$$
$$\omega(R-d) = v_l \,.$$

We can rearrange these two equations to solve for ω the rate of rotation about the ICC and R the distance from the center of the robot to the ICC

$$\omega = (v_r - v_l)/2d$$
$$R = d(v_r + v_l)/(v_r - v_l) \,.$$

The instantaneous velocity of the point midway between the robot's wheels is given by $V = \omega R$.

Now as v_l and v_r are functions of time we can generate the equation of motion for the differential drive robot. Using the point midway between the wheels as the origin of the robot, and writing θ as the orientation of the robot with respect to the x-axis of a global Cartesian coordinate system obtains

$$x(t) = \int V(t)\cos\left[\theta(t)\right] \mathrm{d}t \,,$$
$$y(t) = \int V(t)\sin\left[\theta(t)\right] \mathrm{d}t \,,$$
$$\theta(t) = \int \omega(t)\,\mathrm{d}t \,.$$

This is the solution for the odometry of a differential drive vehicle on the plane. Given control inputs (v_l and v_r) and some initial state estimate, we can estimate where an idealized robot using this motion model will be at any time t.

Given such a model and complete knowledge of the control inputs, we should, in principle, be able to estimate a robot's pose at any time. In a perfect world this would be all that is necessary to estimate accurately the robot's pose at any time in the future. Unfortunately errors in the modelling (incorrect estimations of wheel size, vehicle size), uncertainty about the control inputs, realities of the motor controller (errors between commanded wheel rotation and true rotation), errors in the physical modelling of the robot (wheel compaction, ground compaction, wheel slippage, non-zero tire width), etc., introduces an error between the dead reckoning estimate of the vehicle motion and its true motion. The problem of correcting for this error is the problem of *pose maintenance* for the vehicle, and requires the integration of the dead reckoning estimate with estimates obtained from other sensor systems.

Other chapters in this handbook examine sensors that rely on external events, visual and otherwise, that can provide information as to the robot's pose or changes in it's pose. Here we consider sensors that transduce physical properties of matter under the influence of external forces and properties of matter and the use of global positioning systems (GPS).

Avidbots Ex1016
Page 773 of 2154

29.2 Gyroscopic Systems

The goal of gyroscopic systems is to measure changes in vehicle orientation by taking advantage of physical laws that produce predictable effects under rotation. A rotating frame is not an inertial frame, and thus many physical systems will appear to behave in an apparently *non-Newtonian* manner if the frame is rotating. By measuring these deviations from what would be expected in a Newtonian frame the underlying self-rotation can be determined.

29.2.1 Mechanical Systems

Mechanical gyroscopes and gyrocompasses have a long history in navigation, *Bohnenberger* is generally credited with the first recorded construction of a gyroscope [29.4]. The gyrocompass was patented in 1885 by Martinus Gerardus ven den Bos. In 1903 Herman Anschütz-Kaempfe constructed a working gyrocompass and obtained a patent on the design. In 1908 Elmer Sperry patented a gyrocompass in the US and then attempted to sell this device to the German Navy. A patent battle followed and Albert Einstein testified in the case. ([29.5–8] for more details on the history of the gyrocompass and its inventors.)

Gyroscopes and gyrocompasses rely on the principle of the conservation of angular momentum [29.9]. Angular momentum is the tendency of a rotating object to keep rotating at the same angular speed about the same axis of rotation in the absence of an external torque. The angular momentum $\boldsymbol{L}$ of an object with moment of inertia $\boldsymbol{I}$ rotating at angular speed ω is given by

$$\boldsymbol{L} = \boldsymbol{I} \times \boldsymbol{\omega} .$$

Consider a rapidly spinning wheel mounted on a shaft so that it is free to change its axis of rotation (Fig. 29.2a). Assuming no friction on the bearings or air resistance, the rotor axis will remain constant regardless of the motion of the external cage. This constancy of orientation can be exploited to maintain a bearing independently of the motion of the vehicle, although it is not usually desirable to utilize the principle of conservation of angular momentum via a gyroscope directly. To see this, suppose that a gyroscope is set on the equator, with its spinning axis aligned pointing along the equator (Fig. 29.2b). As the earth spins, the gyroscope will maintain a constant axis of orientation and thus to an earth-fixed observer will appear to rotate returning to its original orientation every 24 hours. Similarly, if the gyroscope were to be positioned on the equator such that its spinning axis was parallel to the axis of rotation of the earth, the gyroscope's axis of rotation would remain stationary and would appear to remain stationary to an earth-fixed observer as the planet rotates.

Although this global motion limits the mechanical gyroscope's ability to sense absolute orientation directly, gyroscopes can be used to measure local changes in orientation, and thus are well suited to vehicular robotic applications. Rate gyros (RGs) measure a vehicle's rotation rate (its *angular rate* of rotation). This is the fundamental measurement that is the basis of all gyroscopic systems. Rate-integrating gyros (RIGs) use embedded processing to internally integrate the angular rotation rate to produce an estimate of the absolute angular displacement of the vehicle.

In order to exploit a gyroscope for navigation with respect to an earth-stable frame, it is desirable for the rotational axis of the shaft to remain fixed within the earth frame, rather than remaining fixed with respect to an external frame. A gyrocompass obtains this by relying on precession. When a torque is applied to change the axis of rotation of a spinning object, conservation of angular momentum will cause a change in the spin direction that is perpendicular to the angular momentum and the direction in which the torque is applied. This is the effect that causes gyroscopes suspended at one end to *spin* around the end from which they have been suspended. Consider the *pendulus gyro* sketched in Fig. 29.3a. This is a standard gyroscope with a weight suspended below the rotational axis. As before, imagine this pendulus gyrocompass set spinning on the equator with the axis of rotation aligned with the axis of rotation of the planet, and with the weight hanging directly down. As the planet spins, the gyroscope's axis of rotation remains stationary and would appear to remain

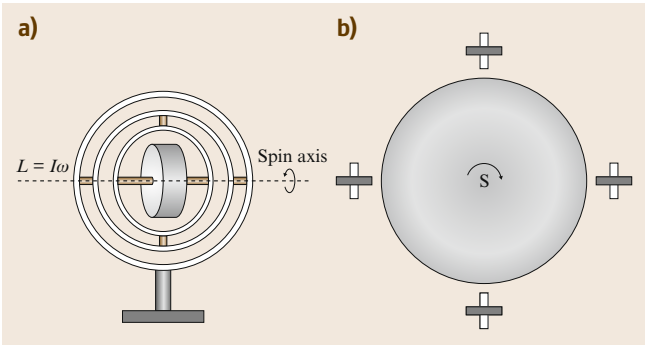


Fig.29.2a,b Mechanical gyroscope. **(a)** Traditionally gimballed gyroscope. The gimball provides the gyroscope the freedom to rotate about its axis as the base of the gyroscope is rotated **(b)** gyroscope as it is rotated around the planet. The wheel of the gyroscope (*grey*) remains in the same orientation as it revolves with the planet. To an observer on the planet the gyroscope will appear to rotate

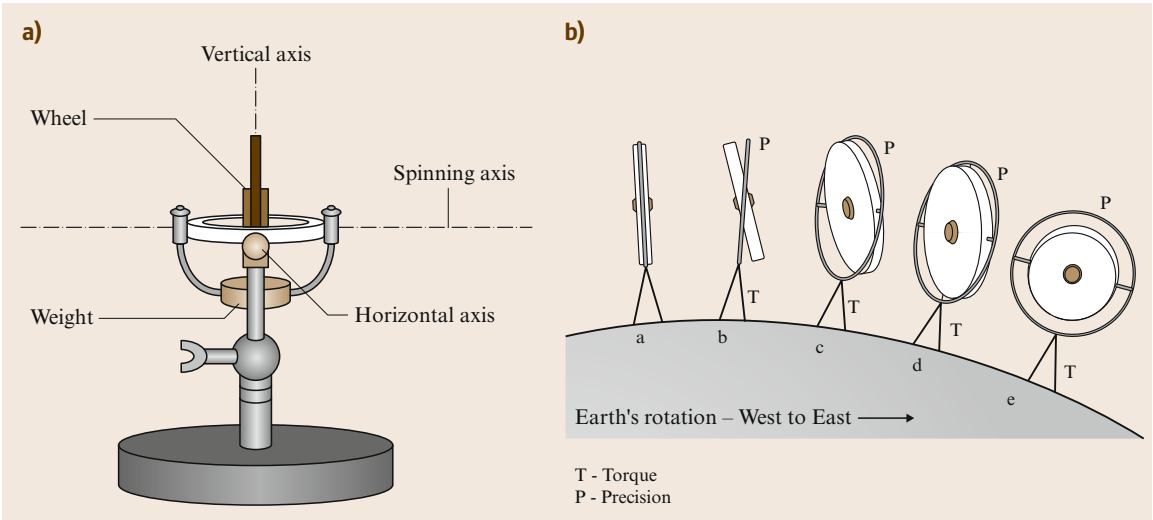


Fig.29.3a,b Simple gyrocompass: **(a)** Pendulus gyro; **(b)** Precessional motion

stationary as the planet rotates. Now suppose that instead of being aligned with the spin axis of the planet, the spin axis is aligned with the equator. As the planet spins, the spin axis is drawn out of the plane as it remains aligned with the original spin axis. As it becomes drawn out of the plane, the mass hanging below the gyrocompass is raised up and generates a torque down due to gravity. The direction perpendicular to the spin axis and the torque rotates the spin axis away from the equator (aka towards the true pole). This process is sketched in Fig. 29.3b.

Unfortunately the pendulus gyro is not an ideal device for navigation. Although its rotation axis will align with the rotation axis of the planet, it does not converge to this value directly but oscillates about it. The solution to this damping problem is to use oil reservoirs, rather than a solid weight, as the counterbalance, and to restrict the motion of the oil in the reservoir [29.10].

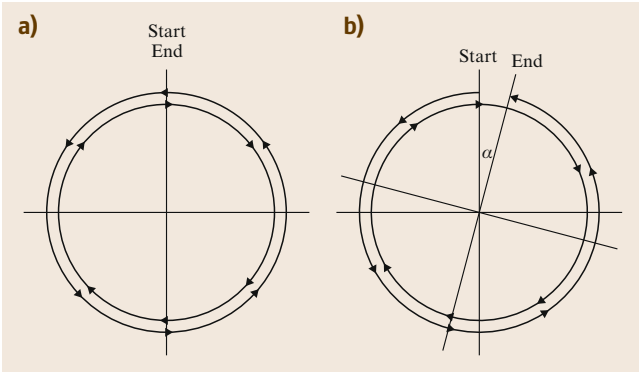


Fig.29.4a,b Circular light path. **(a)** Stationary path; **(b)** Moving path

The gyrocompass finds true north by controlling the precession of a gyroscope. In practice, the performance of a mechanical gyrocompass is impacted by external forces acting on the compass which also contribute to the precession of the gyroscope. This includes forces generated by motion of the device containing the gyrocompass, as well as any external forces acting on the vehicle. Another issue for mechanical gyrocompasses is that in latitudes away from the equator, the stable position of the gyrocompass is not horizontal, and accurate estimates of true north in such latitudes requires corrections to be applied to raw gyrocompass values. Finally, mechanical gyrocompasses require an external force to be applied to maintain the spin of the gyroscope. This process introduces unwanted forces into the system which can further corrupt the measurement process.

Given the complexity, cost, size, the delicate nature of gyrocompasses, and the availability of less expensive and more reliable technologies, mechanical gyrocompasses have given way to optical- and MEMS-based systems (MEMS or *microelectromechanical systems* are also known as *micromachines* or *micro systems technology* and refer to mechanisms at the scale of integrated circuits).

29.2.2 Optical Systems

Optical gyroscopes rely on the *Sagnac effect* rather than rotational inertia in order to measure (relative) heading. The mechanism is based on the behaviour of an optical standing wave in a rotating frame. Historically this was first produced using lasers and an arrangement of mirrors, but it is now typically obtained using fibre-

optic technology. The Sagnac Effect is named after its discoverer *Georges Sagnac* [29.11, 12]. The underlying concept can be traced back even earlier to the work of *Harress* [29.13], and perhaps finds its most famous application in terms of the measurement of the rotation of the earth [29.14].

Ignore relativistic effects and consider the circular light path shown in Fig. 29.4a. If two light pulses are sent in opposite directions around a stationary path of perimeter $D = 2\pi R$ they will travel the same distance at the same speed. They will arrive at the starting point simultaneously, taking time $t = D/c$ where c is the speed of light in the medium. Now let us suppose that instead of being stationary, this circular light path rotates clockwise about its center at rotational speed ω (Fig. 29.4b). The light travelling clockwise around the path must go *farther* to reach the starting point, while light travelling counterclockwise around the path goes a *shorter* distance. The clockwise path has distance

$$D_c = 2\pi R + \omega R t_c \,,$$

where t_c is the time taken in the clockwise direction, while the counterclockwise path has distance

$$D_a = 2\pi R - \omega R t_a \,,$$

where t_a is the time taken in the counterclockwise direction. But $D_c = ct_c$ and $D_a = ct_a$, so

$$t_c = 2\pi \frac{R}{c - \omega R}$$

and

$$t_a = 2\pi R(c + \omega R) \,.$$

The time difference $\Delta t = t_c - t_a$ is given by

$$\Delta t = 2\pi R \left(\frac{1}{c - \omega R} - \frac{1}{c + \omega R} \right) .$$

By measuring Δt, the rotational speed can be computed. Note that although the above derivation assumes classical mechanics and ignores relativistic effects, the derivation also applies when relativistic speeds are taken into account [29.15]. See [29.16] for an in depth review of the Sagnac effect and ring-lasers.

In optical gyroscopes lasers are typically used as the light source. Optical gyroscopes either utilize straight line light paths with mirror surfaces or prisms at the edges to direct the light beam (a ring laser gyroscope – RLG), or a polarization maintaining glass-fiber loop (fiber optic gyro – FOG). The glass-fiber may actually loop multiple times, thus extending the effective length of the light path. The time delay between the clockwise and counter-clockwise is detected by examining the phase interference between the clockwise and counter-clockwise light signals. Multiple optical gyroscopes with non-parallel axes can be ganged together in order to measure 3-D rotations.

Various techniques can be used to measure the time difference including examining the Doppler (frequency) shift of the laser light due to the motion of the gyro and an examination of the beat frequency of the interference pattern between the clockwise and counter-clockwise paths [29.17]. Ring interferometers typically consist of many windings of fiber optic lines that conduct light of a fixed frequency in opposite directions around the loop and measure the phase difference. A ring laser consists of a laser cavity in the shape of a ring. Light circulates in both directions around this cavity producing two standing waves with the same number of nodes in both directions. Since the optical path lengths are different in the two directions, the resonate frequencies differ. The difference between these two frequencies is measured. An unfortunate side effect of the ring laser approach is that the two signals will lock to each other for small rotations and it is typically necessary to physically rotate the device in a controlled manner in order to ensure that this lock-in effect can be avoided.

29.2.3 MEMS

Almost all MEMS gyroscopes are based on vibrating mechanical elements to sense rotation. Vibratory gyroscopes rely on the transfer of energy between vibratory modes based on Coriolis acceleration. Coriolis acceleration is the apparent acceleration that arises in a rotating frame of references. Suppose an object moves along a straight line in a rotating frame of reference. To an outside observer in an inertial frame the object's path is curved – thus there must be some force acting on the object to maintain the straight line motion as viewed by the rotating observer. An object moving in a straight line with local velocity $\boldsymbol{v}$ in a frame rotating at rate $\boldsymbol{\Omega}$ relative to an inertial frame will experience a Coriolis acceleration given by

$$\boldsymbol{a} = 2\boldsymbol{v} \times \boldsymbol{\Omega} \,.$$

Transducing acceleration in a MEMS gyroscope amounts to inducing some local linear velocity and measuring the resultant Coriolis forces.

Early MEMS gyroscopes utilized vibrating quartz crystals to generate the necessary linear motion. More

Avidbots Ex1016
Page 776 of 2154

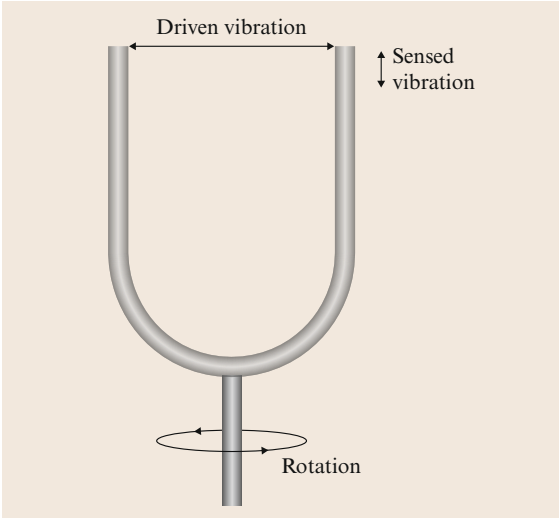


Fig. 29.5 MEMS gyroscope principle of operation

recent designs have replaced the vibrating quartz crystals with silicon based vibrators. Various MEMS structures have been developed including

Tuning Fork Gyroscopes

Tuning fork gyroscopes use a tuning fork-like structure (Fig. 29.5) as the underlying mechanism. As the tuning forks vibrate within a rotating frame, Coriolis forces cause the tines of the fork to vibrate out of the plane of the fork which is measured. This is the effect used by the InertiaCube sensor [29.18].

Vibrating Wheel Gyroscopes

Vibrating wheel gyroscopes use a wheel that oscillates about its rotational axis. External rotation of the frame causes the wheel to tilt which can be sensed.

Wine Glass Resonator Gyroscopes

Wine glass resonator gyroscopes use the effect of Coriolis forces on the position of nodal points on a resonating structure to estimate the external rotation. Sadly, no actual wine glasses are involved.

As MEMS gyroscopes have no rotating parts, have low power consumption requirements, and are very small, MEMS gyros are quickly replacing mechanical and optical gyroscope sensors in robotic applications.

With the exception of gyrocompasses, gyroscopes measure relative rotational motion around a single axis. They accomplish this measurement by exploiting physical properties of rotating frames of reference. Earlier technologies based on mechanical gyroscopes have given way to optical- and MEMS-based devices but the underlying principle remains unchanged – that rotating frames of reference show specific physical properties can be measured to estimate the relative rotation.

A problem common to all gyroscopes is that of drift. Each of the relative motion measurements is corrupted by an error process, and these errors accumulate over time. This, coupled with specific measurement errors associated with the individual gyroscope technologies means that unless the error is corrected through reference to some alternate (external) measurement, the drift will eventually exceed the required accuracy of the measurement.

As individual gyros only measure rotation about a single axis, it is common to gang multiple gyros together with orthogonal axes of sensitivity in order to measure 3-D rotations. These collections of gyros are often integrated with other sensors (compasses, accelerometers, etc.) in order to construct *inertial measurement units* (or IMUs). This is considered in Sect. 29.4.

29.3 Accelerometers

Just as gyroscopes can be used to measure changes in orientation of a robot, other inertial sensors – known as *accelerometers* – can be used to measure external forces acting on the vehicle. One important factor concerning accelerometers is that they are sensitive to all external forces acting upon them – including gravity. Accelerometers use one of a number of different mechanisms to transduce external forces into a computer readable signal.

29.3.1 Mechanical Accelerometer

A mechanical accelerometer (Fig. 29.6a) is essentially a spring-mass-damper system with some mechanism for external monitoring. When some force is applied (e.g., gravity), the force acts on the mass and displaces the spring. Assuming an ideal spring with a force proportional to its displacement, the external forces balance the internal ones

$$\begin{aligned} F_{\text{applied}} &= F_{\text{intertial}} + F_{\text{damping}} + F_{\text{spring}} \,, \\ &= m\ddot{x} + c\dot{x} + kx \,, \end{aligned}$$

where c is the damping coefficient. This equation can be solved to show that depending on the size of the damping coefficient relative to the expected external force and the mass, the system can be made to reach a stable final value in a reasonably short period of time when-

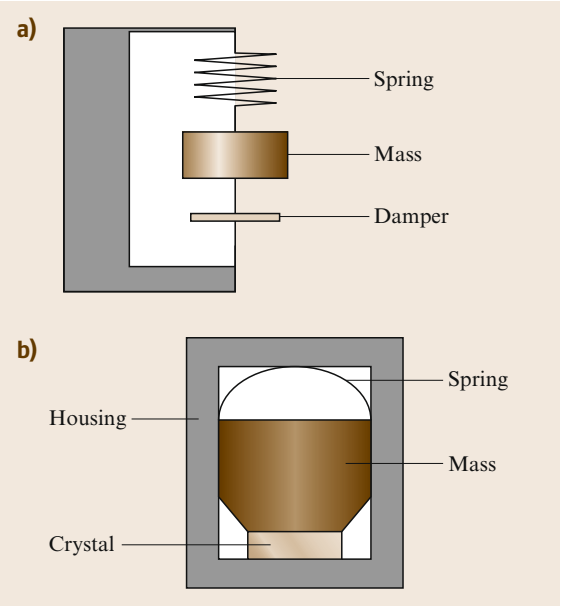


Fig. 29.6 Accelerometers. **(a)** Mechanical accelerometer; **(b)** piezoelectric accelerometer ◀

ever a static force is presented. This need to pre-estimate the expected force and the resulting (potentially long) time for the system to converge on a final measurement coupled with non-ideal performance of the spring limits the applicability of mechanical accelerometers. Another issue with mechanical accelerometers is that they are particularly sensitive to vibration.

29.3.2 Piezoelectric Accelerometer

Rather than relying on a direct mechanical measurement of external forces, piezoelectric accelerometers are based on a property exhibited by certain crystals which generate a voltage across them when stressed. A small mass can be positioned so that it is only supported by the crystal, and as forces cause the mass to act upon the crystal this induces a voltage that can be measured (Fig. 29.6b).

29.4 IMU Packages

An inertial measurement unit (IMU) is a device that utilizes measurement systems such as gyroscopes and accelerometers to estimate the relative position, velocity and acceleration of a vehicle in motion. First demonstrated in 1949 by C. S. Draper, IMU's have become a common navigational component of aircraft and ships. Historically an IMU is self-contained and provides this estimate without reference to external references, however the definition has become less precise in recent years and now it is common to have the term IMU also refer to systems which do include such external references.

IMU's come in two basic flavours, gimbaled systems and strap-down systems. As their name suggests, gimbaled IMU's are mounted within complex gimball structures in order to provide a stable platform from which measurements can be made. Gyroscopes are used to ensure that the gimball remains aligned with the initial reference frame at power up. The orientation of the gimbaled platform relative to the vehicle is used to map measurements taken within the IMU to the reference frame of the vehicle. Strap down IMU's, on the other hand, have the IMU rigidly connected to the vehicle (strapped down), so no such transformation is required. In either case estimating the motion relative to the initial frame requires integrating information from the sensors within the IMU (accelerometers, gyroscopes, etc.) in real time. This was a significant computational effort in the early days of IMU's, and thus historically (prior to the 1970s) the gimbaled IMU was more common. Given the low expense of such computation today, and the costs associated with manufacturing and operating gimbaled IMU's, strap down IMU's are much more common today [29.19].

The sensing technology of an IMU is typically integrated with an on-board computational unit to produce an attitude and heading reference unit (AHRS) which is a single device that maintains a 6-DOF estimate of the pose of the vehicle (position (x, y, z) and orientation (roll, pitch, yaw)). In practice, the distinction between an AHRS and an IMU is often unclear since the combinations of software and device outputs is quite varied, and most MEMS devices contain a substantial computational substrate and are able to directly offer up a position, velocity, and attitude (PVA) solution. In general, an AHRS includes gyroscopes, accelerometers and a magnetometer. These are often combined with a GPS as described in the next section. In addition to maintaining a 6-DOF pose of the vehicle, commercial IMU's also typically maintain estimates of velocity and acceleration and can usually provide a diverse set of lower-level measurements as well.

The basic computational task of an IMU is shown in Fig. 29.7. This IMU uses three orthogonal accelerometers and three orthogonal gyroscopes. The Gyroscope data $\boldsymbol{\omega}$ is integrated to maintain an ongoing estimate

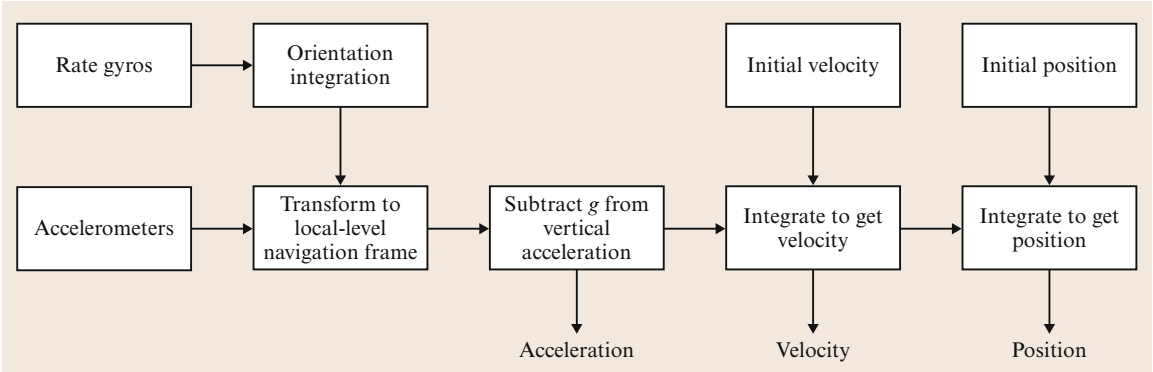


Fig. 29.7 IMU block diagram

of vehicle orientation $\boldsymbol{\theta}$. At the same time, three accelerometers are used to estimate the instantaneous vehicle acceleration $\boldsymbol{a}$. This data is then transformed via the current estimate of the vehicle orientation relative to gravity, so that the gravity vector can be estimated and extracted from the measurement. The resulting acceleration is then integrated to obtain vehicle velocity $\boldsymbol{v}$ and then integrated again to obtain position $\boldsymbol{r}$.

IMU's are extremely sensitive to measurement errors in the underlying gyroscopes and accelerometers. Drift in the gyroscopes leads to mis-estimates of the vehicle orientation relative to gravity, resulting in an incorrect cancellation of the gravity vector. As the accelerometer data is integrated twice, any residual gravity vector will result in a quadratic error in position [29.18]. As it is never possible to completely eliminate the gravity vector and this and any other error is integrated over time drift is a fundamental issue for any IMU. Given a sufficiently long period of operation all IMU's eventually drift and reference to some external measurement is required to correct this. For many field robots GPS has become an effective source for these external corrections.

29.4.1 Performance

Gyroscopes and Inertial measurements units today frequently include the required computational support to directly integrate the inertial signals and provide position, velocity, and attitude (PVA) estimates as direct output. The quality of these estimates is critically dependent not only on the sensing technology, but on the algorithms (almost universally including an extended Kalman filter) used in the estimation process and their calibration. While the extended Kalman filer is, of course, a well understood algorithm, the subtlety of its implementation remains a significant distinguishing factor among different brands of IMU.

Intertial measurement units can be evaluated with respect to various factors that determine performance. Several of these are enumerated below:

1. Bias repeatability. This is maximum deviation of the gyroscope under fixed inertial operation conditions with constant temperature, i. e., the drift of the reading under ideal conditions. This is measured over different time scales and leads to *short term* and *long term* bias repeatability.
2. Angle random walk. This measures the noise in the angular rate data coming from the gyro.
3. Scale factor ratio. This parameter is not specific to IMU's or gyros and is a general measurement of signal amplitude. It measures the ratio of the output analogue voltage to the sensor parameter of interest. For a gyroscope this is typically measured in $\mathrm{mV/(deg/s)}$ whereas for an accelerometer it is typically $\mathrm{mV/(m/s^2)}$.
4. Position accuracy. The accuracy in practice of position estimation depends on the sensors and the integration algorithm, both of which are intrinsic to the IMU, as well as the nature of the trajectory experienced by the IMU.

29.5 Satellite-Based Positioning (GPS and GLS)

The global navigation satellite system (GNSS) and it's most commonplace instance, the global positioning system (GPS), is the single most commonly used mechanism for location estimation. It provides a three dimensional position estimate in absolute coordinates as well as current time and date and is available any-

where on the earth's surface. Standard GPS provides a position estimate in the horizontal plane to within about 20 meters. It was original developed for military applications but has become widely adopted in civilian applications including automobile navigation systems, recreational orienteering, and inventory tracking for transportation companies. It is rapidly being superseded (via augmentation) by systems that add related technologies to basic GPS reception.

29.5.1 Overview

GPS is based on received radio signals transmitted by an ensemble of satellites orbiting the earth. By comparing the time delays from the different satellite signals, a position fix can be computed. The most widely accepted GPS system is based on the NAVSTAR satellite system deployed and maintained by the United States, specifically by the Air Force Space Command. As a US military service, the US government reserves the right to terminate or modify it's availability at their discretion. A similar system named GLONASS (globalnaya navigatsionnaya sputnikovaya sistema) is operated by the Russian government. Many modern GPS receivers and cell phones can access both the US GPS and Russian GLONASS systems. Another alternative system is being deployed by the European Union, named *Galileo*, with the explicit expectation that it not be under military control. It is expected to offer two different classes of service, an open service and an encrypted higher-quality commercial service. Other GPS systems such as the Chinese Beidou, the Japanese quasi-zenith satellite system (QZSS), the Indian regional navigational satellite system (IRNSS) system are intended to provide similar functionality. GPS receivers that are able to receive position data from multiple different satellite systems have become readily available and are referred to as global navigation satellite system (GNSS) systems.

Technically, the term GPS always refers to the NAVSTAR system although in common parlance it is sometime used to refer to any system that provides satellite-based positioning. Historically NAVSTAR provided two different services: the precise positioning system (PPS), reserved primarily for military users, and the standard position system (SPS) with lower accuracy. The difference between these was referred to as selective availability (SA). This difference in accuracy was artificially induced via pseudo-random noise in the SPS signal for strategic reasons, and was eliminated on May 2, 2000. Although this accuracy distinction could, in principle, be reinstated this seems unlikely, and recent GPS satellites (GPS Block III series, not launched at this time) are not equipped with the SA feature. US military users have access to a precise positioning service (PPS). This second frequency enables military users to correct for radio signal degradation caused by the Earth's atmosphere.

The GPS satellite network is based on a base constellation of 24 orbiting satellites along with up to six supplementary additional satellites that are also operational. These satellites are in almost-circular medium earth orbit. As opposed to being geostationary the orbits are semi-synchronous, meaning that their position relative to a ground observer is constantly changing, and that their orbital period is exactly half a sidereal day. The orbits are selected so that from almost any point of the earth's surface there will always be four or more satellites directly visible – a criterion for obtaining a GPS position estimate. The satellites are organized into six orbital planes with four satellites in each. The system is designed such that averaged over the entire earth's surface and over a 24 hour internal, the satellites should provide 99.9% coverage at the worst-covered location in any 24 hour interval, and the signal should be at least 83.92% available at the worst place on earth on the worst day over a 30 day measurement interval. Note that this criterion for availability takes into account transmitted operational factors above and beyond simple coverage. Of course, this criterion ignores the reality of topographic features like mountains, as well as other objects such as buildings that can obstruct the like-of-sight.

Each satellite repeatedly broadcasts a data packet known as the coarse-acquisition (C/A) code, which is received by the GPS receiver on the L1 channel at 1575.42 MHz. The simple principle is that if the receiver knows the absolute positions of the observed satellites, the receiver position can be directly determined. If the signal propagation time for the radio signals were known, the receiver position could be computed directly via trilateration (Fig. 29.8). This implies that an absolute timing reference is present on the receiver, which would be prohibitively costly. Instead, only the satellites have highly accurate atomic clocks. The receiver computes the difference in signal propagation times between the different satellites, and uses this to compute a range estimate referred to as a *pseudo-range* (to explicitly indicate that it is corrupted by several sources of measurement noise). The specific geometric problem is referred to as *multilateration* or *hyperbolic positioning* and the solution is computed using a sophistical Kalman filter within the GPS receiver. To avoid retaining an ephemeris (pose) table for the satellites and a very accurate clock in the receiver, each satellite broadcasts its own position and an accurate time signal as part of the data packet that it transmits.

GPS satellites broadcast at several different frequencies known as L1 through L5 although until recently only L1 (1575.42 MHz) and L2 (1227.6 MHz) were being using for civilian GPS receivers. Dual frequency L1/L5 GPS systems have become more broadly largely due to the emergence other GNSS systems in addition to the original NAVSTAR GPS.

The standard service offered by NAVSTAR and the performance criteria for it are determined by the L1 signal, which contains two unencrypted components: the acquisition message (coarse acquisition message C/A) and a navigation data message. It is also possible to use the encrypted L2 signal as well, even without the secret decryption keys, to provide augmented error correction (by observing the relative effects of ionospheric distortion as a function of frequency). The restricted access signal broadcast on both the L1 and L2 channels is the P-code (as well as a fairly recent M-code) which, is known as the Y-code or P(Y) or P/Y code once it is encrypted. Both the C/A and P(Y) code include the navigation message stream that specifies clock bias data, orbital information, ionospherical propagation corrections factors, ephemeris data, status information on all the satellites, universal time code and other information. The satellite performance is coordinated by the master control station located at the Schriever Air Force Base near Colorado Springs, (Colorado, USA) and is connected to a global network of five additional monitoring stations (Cape Canaveral, Ascension Island, Kwajalein Atoll (Marshall Islands), Diego Garcia Atoll, Hawaii) which are used to produce the measurements that are uplinked to generate the navigation message stream. Finally, it should be noted an additional signal is now available on the L2 frequency band. This L2C signal on satellites designated *block IIR-M* promises to provide much improved receiver sensitivity so that position fixes can be obtained in environments, such as in forests, where they are currently not readily available.

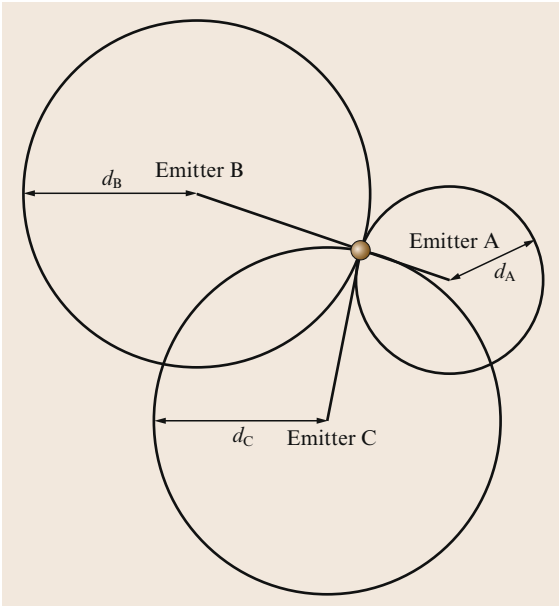


Fig. 29.8 GPS trilateration on the plane. Suppose that one receives signals from three transmitters (A, B, and C) with known locations. Knowledge of the signal delay from one emitter (say A) localizes the receiver to lie on a circle of known diameter (d_A) whose center is the emitter. The constraints from two emitters intersect at two points (maximum). A third emitter is required to disambiguate these two solutions. In 3-D, the signal propagation constraint from a single emitter constrains the receiver to lie on a sphere. The intersection of the constraints from two emitters constrains the receiver to lie on a circle. The intersection of the constraints from three emitters constrains the receiver to one of two points

29.5.2 Performance Factors

GPS performance depends on several factors: satellite transmission accuracy, environmental conditions, interactions with ground-based obstacles, and receiver properties.

In the context of robotics, factors that effect the performance of the satellites themselves and the atmospheric conditions are essentially uncontrollable. Nevertheless, it should be noted that these can be sources of error and that the GPS signal itself may not always be reliable. A *service failure* is defined as a set of circumstances where the positioning service exhibits atypical error behavior (i. e., incorrect signals). Such failures are classified into minor and major failures. Minor failures are those that have limited impact on a receiver and lead to ranging error of under 150 m. Major failures are those that lead either to larger errors or data processing overloads in the receiver. If a single satellite experiences an error that leads to a major failure, then within a 6 hour period, approximately 63% of the earth's surface will have the satellite in view a some point.

The controllable factors in using GPS for accurate localization are (i) it requires an unobstructed line of sight to the satellites, (ii) it depends on atmospheric conditions, and (iii) it depends the ability to receive (weak) radio-frequency communications. There is a potential for wildly incorrect estimates. Generally satellites that are directly overhead provide better signal than those near the horizon. In addition, since the basis of GPS position is differential signal analysis, it is best if the satellites used in the GPS computation are widely spaced in the sky.

Avidbots Ex1016
Page 781 of 2154

GPS signals are in the microwave band and, as such, they can pass through plastic and glass, but are absorbed by water (wood, heavy foliage) and are reflected by many materials. As a consequence, GPS is unreliable in heavy forest, deep canyons, inside automobiles and boats, heavy snowfall or between tall buildings. In some cases, partial obstruction of the sky may not prevent a position estimate from being computed. Assuming the minimum number of satellites operating at any time is 24, then on average across the earth's surface 8 satellites are in view so that even partial occlusion of the sky can often be tolerated. On the other hand, partial occlusion can lead to reduced accuracy since the selection of available satellites used for computing position becomes limited and optimal accuracy is obtained by using as many satellites as possible (weighting them appropriately in the internal Kalman filter).

Secondary factors that differentiate different GPS receivers are the rate at which the signals are collected, the receiver sensitivity, the number of satellites used in the final computation, the number of factors taken into account in the estimator and the exploitation of supplementary positioning schemes such as WAAS (described later). A major factor in determining the rate at which estimates can be produced is the number of independent receiver elements in the GPS system. Sequential single-channel receivers are simpler and thus more economical (and potentially smaller), but they must sequentially lock onto each satellite being used. Parallel multi-channel receivers can lock onto to more than one satellite at once, and are generally faster and more costly; some degree of parallelism is the norm in good-quality consumer devices.

GPS computations are based on an estimation of the dilution of precision (DOP) and specifically for the dilution of precision of the positional parts of the system (i. e., positional dilution of precision PDOP). These correspond to partial derivatives of the error with respect to position and allow the most accurate ensemble of visible satellites to be determined at any time. The standard implementation for GPS systems specifies that PDOP values be recomputed every 5 minutes.

The minimum performance parameters for GPS receivers are based upon transforming instantaneous range residuals to a user position estimate using a linearized position solution from a stationary, surveyed location. Most GPS receivers use additional techniques such as range residual smoothing, velocity aiding, Kalman filtering, or multiple satellite (*all-in-view satellite*) solutions. That said, formal performance for the system is measured with respect to the minimum. The GPS position estimation algorithm is summarized as follows:

1. Select the best four satellites based upon minimum error measured in terms of PDOP.
2. Update every five minutes, or whenever a satellite being used in the solution sets.
3. Measure the pseudo range to each satellite. Each of the four measurements must have a reception timetag within 0.5 s of the solution time. The reception timetag is based upon measurement system time, and the transmission timetag is based upon satellite time.
4. Determine the ephemeris for each of the satellites being used, and compute the Earth-centred, Earth-fixed (ECEF) coordinates for each. Correct for the earth's rotation and thus compute an estimate pseudo-range measurement that should be obtained for each satellite.
5. Compute the *range residuals* as the differences between the actual and observed measurements.
6. Estimate the matrix $\mathbf{G}$ that determines the overall system solution, know as the *position solution geometry matrix*. The matrix can be described in terms of a collection of row vectors one for each of the satellites being used, each row being made up of the x, y, z and time coordinate direction cosines for the vector between the user and the satellite (the is with respect to a fixed reference frame for the planet called the World Geodetic System, WGS84). WGS84 represents position on the Earth in terms of latitude, longitude, and height above the surface, which is modeled as an ellipsoid. The WGS84 coorinate system represents longitude as a value from −180 to 180 degrees, with 0 degrees at the Earth's prime meridian, and longitude as a value from 90 degrees (at the North pole) to −90.
7. Compute the user position.

The standard implementation of GPS is based upon a position fix rate of once per second, although faster and slower rates are possible. Under typical operating conditions and without specialized enhancements GPS accuracy is roughly 20−25 m in the horizontal plane and 43 m in the vertical direction. The restricted PPS signal provides an accuracy of at least 22 m (typical values are 7−10 m) in the horizontal plane and 27.7 m in the vertical direction as well as UTC time accuracy within 200 ns based on a reference signal from the US Naval Observatory.

GPS signals can be affected by multipath issues, where the radio signals reflect off surrounding terrain – buildings, canyon walls, hard ground, etc. This delay in reaching the receiver causes inaccuracy. A variety of receiver techniques, most notably narrow correlator spacing [29.20], have been developed to mitigate multipath errors. For long delay multipath, the receiver itself

can recognize the wayward signal and discard it. To address shorter delay multipath due to the signal reflecting off the ground, specialized antennas may be used. This form of multipath is harder to filter out as it is only slightly delayed as compared to the direct signal, causing effects almost indistinguishable from routine fluctuations in atmospheric delay.

29.5.3 Enhanced GPS

A wide range of augmentation systems have been developed both by government agencies and private companies to enhance the basic GPS signal.

Wide Area Augmentation System (WAAS)

The wide area augmentation system (WAAS) is an augmentation system operated by the US Federal Aviation Administration (FAA) to provide better GPS and timing performance for aircraft operating over North America. The WAAS system is based around a supplementary signal that can be received by GPS receivers to improve their accuracy. WAAS increases the accuracy of horizontal position estimates from 10–12 m with GPS alone, to between 1 and 2 m. The WAAS signal contains corrections for the GPS signal that reduce the effects of errors due to timing errors, satellite position corrections, and local perturbations due to variations in the ionosphere. These correction terms are estimated by ground-based stations at fixed and accurately-known positions and uplinked to satellites which broadcast them to suitably enabled GPS receivers. The WAAS signal is only computed and available for North America, but similar correction signals are becoming available elsewhere as part of the standardization of satellite-based augmentation systems (SBASs). This includes Europe (where is it called EGNOS, the European Geostationary Navigation Overlay Service) and Japan and parts of Asia (where it is called MSAS, the multifunctional satellite augmentation system). Further enhancements to GPS and WAAS, in the form of global navigation satellite system landing system (GLS), are slated for completion in 2013.

Differential GPS

Differential GPS (DGPS) is a technique for correcting GPS signals by using a nearby GPS receiver located at a known accurately-surveyed position. In fact, several variations on this basic idea exist and are also known under the general rubric of ground based augmentation systems (GBASs). DGPS uses the same principles as WAAS but on a local scale without resorting to the use of satellite uplinks. The receiver at the known position computes the error in the GPS signal and transmits it to the nearby receiver at the unknown location. Since the error varies as a function of position on the earth, the effectiveness of the correction degrades with distance, typically with a maximum effective range of a couple of hundred miles. The method was especially desirable before the suspension of selective availability and the development of WAAS (which can be viewed as a form of DGPS). In the USA and Canada, a network of ground based DGPS transmitters are in place, send signals using radio frequencies between 285 and 325 kHz. Commercial DGPS solutions also exist akin to WAAS.

29.5.4 Nationwide Differential GPS System (NDGPS)

The (US) nationwide differential GPS system (NDGPS) is an augmentation system that integrates costal and ground transportation systems to provide 10–15 cm accuracy throughout most of the US. The system relies on a network of radio signals broadcast from base stations. Similar systems exist in a number of other countries including Canada.

Receiver Autonomous Integrity Monitor (RAIM)

Receiver autonomous integrity monitor (RAIM) is a technique by which multiple pseudo-range measurements are obtained (i. e., pose estimates) using different combinations of satellites. If inconsistent measurements are obtained, it indicates that a failure of some sort has taken place in the system. A position fix using at least satellites is needed to detect such an error which at least six satellite are needed to exclude the data from a single bad satellite and still obtain a reliable estimate.

Real-Time Kinematic Positioning (RTK)

One GPS enhancement technology with rapidly growing acceptance in robotics applications is real-time kinematic positioning or RTK [29.21]. RTK-GPS in a GNSS configuration also exploits multiple satellite constellations and is often regarded as the most high-performance GPS solution that is readily available. Because of it's high accuracy, it is often referred to as *survey grade*. In practice, commonplace RTK-GPS units can achieve horizontal positioning accuracy on the order of 1 cm. RTK-GPS augments standard differential GPS technology by enhancing the effective resolution of the pseudo-range distance estimates using corrections obtained by a base station. Specifically, RTK estimates the number of carrier cycles in the communication path between a GPS satellite and each of the base station and the receiver, and thus is essentially dependent on the phase of the signal. Such units typically ignore the actual data payload of the signal and exploit the phase of the GPS signal (typically the L1 band); this is a fundamentally different mode of operation from

the commonplace code phase computation that low cost receivers use. The base station that is used to provide a reference position for the estimation process can be part of a custom system being deployed, but various networks reference base stations accessible to the public now exist. When a network of base stations are used to provide correction signals over a wide region, the system is referred to as network real-time kinematic (NRTK) GPS and a class of networks as continuously operating reference station (CORS) networks. NRTK networks are evolving rapidly and a range computing protocols have been developed or employed including the Radio Technical Commission for Maritime Services Special Committee 104 (RTCMS C104) protocol for transmitting GNSS data.

If properly estimated, RTK-GPS can obtain position estimates that are several orders of magnitude more accurate than the multiple meter accuracy associated with conventional DGPS. Critical to RTK is estimating the number of whole signal cycles that are present in the path from the satellite to the station. GPS carrier waves have a wavelength of about 19 cm, and this coupled with an estimate of the phase of the carrier signal can result in pseudo-range estimates with errors in the 1 mm range.

29.5.5 GPS Receivers and Communications

GPS receivers are classified according to their performance and cost. The best receivers are referred to as *geodetic grade* (or survey grade) with economical models referred to as *resource grade* or *recreational*. In general the costs of these different models varies by several orders of magnitude, but the gap in performance is gradually narrowing.

Receivers come in two types: code phase and carrier phase. Code phase receivers use the satellite navigation message part of the data stream to provide the ephemeris data and produce real-time output. There is a delay for them to lock into the satellites, but then they produce output continuously without an initial position estimate. The C/A signal is a 1023 pseudo-random *noise* (PRN) bit string with a known key. The actual pseudo-range data is determined by finding the offset of this bit string, and computing the correct shift of this bit string is the essential computation that determined the delay before a lock is achieved.

Carrier phase receivers, on the other hand, use the phase of the raw GPS signal rather than the embedded (digital) C/A signal. The L1 and L2 signals have wavelengths of 19 and 24 cm respectively, and good quality phase measurements allow horizontal positioning accuracies on the order of millimeters in principle. These measurements, however, only provide relative position information within a neighborhood of some tens of kilometers.

Serial Protocols

Consumer GPS devices almost universally support some variant of the National Marine Electronics Association (NMEA) protocol as well as various proprietary protocols specific to each manufacturer. NMEA is a serial protocol sometimes encapsulated within other protocols. Several variants of the protocol exist, but NMEA 0183 is the most commonly supported while NMEA 2000 supports higher data rates and uses a very different communication protocol. While the protocol is proprietary and the official specification can only be purchased from the NMEA, there are several open-source descriptions of the protocol that have been reversed engineered and its basic message characteristics are very widely used and distributed.

The NMEA 0183 protocol supports an ASCII mode of communications based on a *talker* (the GPS receiver) and one or most *listeners* (computers) which receiver simple protocol strings called sentences. Anecdotal evidence suggests that there are ambiguities in the protocol that can lead to difficulties is assuring compatibility between devices (this suggestion is necessarily made without inspecting the proprietary documentation, which might preclude its description here).

29.6 GPS–IMU Integration

It has become commonplace for GPS and IMU technologies to be integrated into a single GPS-aided inertial navigation system (GPS/INS) to leverage their complementary advantages. Although GPS offers the promise of high resolution positioning information on or about the surface of the planet, it does not solve all of the problems associated with robot pose estimation. First, it does not directly obtain information about vehicle orientation – to determine the orientation of the vehicle (yaw, and for many vehicles pitch and roll) must be estimated by either differentiating the GPS signal or through integration with other sensors including compasses, gyrocompasses, and IMU's. Second, GPS receivers are generally unable to provide continuous independent estimates of position. Estimates are only available at distinct time instances with (for inexpensive receivers at least) considerable delays between measurements. A continuous

estimate of pose requires estimation of pose between GPS readings. Finally, it is not always possible to obtain a GPS fix. Local geography (e.g., mountains, buildings, trees) or an overhead cover that is opaque to radio signals (e.g., indoors, underwater) can block the signal entirely. Integration of a GPS receiver with other sensor technology (often an IMU) can be used to deal with these issues, at least for short periods of time.

The process of integrating GPS and IMU data is typically expressed as an extended Kalman filter estimation process (Sect. 35.2.3). Essentially the IMU data is used to bridge between solid GPS measurements and is combined in a least squares optimal sense with the GPS data when both are available. Given the complementary nature and true independence of the two sensors, a wide range of commercial packages have been developed to integrate GPS and IMU data [29.22].

When GPS measurements are integrated in a robotic vehicle, and in particular when they are combined with data from an IMU the orientation of the antenna becomes a consideration. The GPS position *fix* is based on the position of the GPS antenna and if it is located far from the IMU or the center of the vehicle coordinate system then this must be accounted for or instabilities in the position or orientation can occur.

29.7 Further Reading

29.7.1 Odometry

Many general robotics books including [29.2] and [29.23] provide considerable information on vehicle odometry and derive odometry equations for standard vehicle designs.

29.7.2 Gyroscopic Systems

Everett's book [29.24] provides a review of various sensor technologies including gyroscopic systems and accelerometers. Interesting historical documentation on the gyrocompass and its inventor can be found in *Hughes*' book [29.5].

29.7.3 Accelerometers

Everett's book [29.24] provides a review of various sensor technologies including gyroscopic systems and accelerometers.

29.7.4 GPS

Considerable details on the theory and implementation of GPS systems can be found in *Leick*'s book [29.25]. See also [29.26]. Details of various approaches to GPS/INS integration can be found in [29.27] and [29.28]

29.8 Currently Available Hardware

Although the specific models listed below are likely to have a short shelf life, the list of contacts is may be a good starting point for the identification of specific inertial sensing devices.

29.8.1 Gyroscopic Systems

- KVN DSP-3000 Tactical Grade
  Fiber Optic Gyro (FOG),
  KVH Industries Inc., 50 Enterprise Center,
  Meddletown RI, 02842-5279,
  USA, 1-401-847-3327
- Fiber Optic Gyroscope HOFG-1(A),
  Corporate Headquarters Hitachi Cable Ltd.
  4-14-1 Sotokanda, Chiyoda-ku,
  Tokyo 101-8971, Japan.
- Rate Gyroscope CRS03,
  Silicon Sensing Systems Japan, Ltd.
  1-10 Fusocho (Sumitomo Precision Complex),
  Amagaxaki, Hyogo 660-0891, Japan.

29.8.2 Accelerometers

- Accelerometer GSA 101,
  A-KAST Measurements and Control Ltd.,
  1054-2 Centre St. Suite #299,
  Thornhill, ON, L4J 8E5, Canada.
- ENDEVCO MODEL 22,
  Brüel and Kjaer,
  DK-2850 Naerum, Denmark.

29.8.3 IMU Packages

- μIMU,
  MEMSense, 2693D Commerce Rd.,
  Rapid City, SD 57702, USA.
- IMU400 MEMS Inertial Measurement Unit,
  Crossbow Technology Inc.,
  4145 N. First St., San Jose, CA 95134, USA.
- IntertiaCube3, (3DOF IMU),
  Intersense,
  36 Crosby Dr, #15,
  Bedford, MA 01730, USA.

29.8.4 GPS Components

- Garmin GPS 18,
  Garmin International Inc.,
  1200 East 151st St.,
  Olathe, KS 66062-3426, USA.
- Magellan Meridian Color,
  Thales Navigation
  471 El Camino Real,
  Santa Clara, CA 95050-4300, USA.
- TomTom Bluetooth GPS Receiver,
  Rembrandtplein 35,
  1017 CT Amsterdam,
  The Netherlands.

References

29.1 Vitruvius: *Ten Books on Architecture* (Harvard Univ. Press, London 1914) p. 301, English translation by M. H. Morgan

29.2 G. Dudek, M. Jenkin: *Computational Principles of Mobile Robotics*, 2nd edn. (Cambridge Univ. Press, Cambridge 2010)

29.3 E. Maloney: *Dutton's Navigation and Piloting* (US Naval Institute Press, Annapolis 1985)

29.4 J.G.F. Bohnenberger: Beschreibung einer Maschine zur Erläuterung der Geseze der Umdrehung der Erde um ihre Axe, und der Veränderung der Lage der lezteren, Tüb. Bl. Naturw. Arzneik. **3**, 72–83 (1817)

29.5 T.P. Hughes: *Elmer Sperry: Inventor and Engineer* (The Johns Hopkins Univ. Press, Baltimore 1993)

29.6 H.W. Sorg: From Serson to Draper – Two centuries of gyroscopic development, Navigation **23**, 313–324 (1976)

29.7 W. Davenport: *Gyro! The Life and Times of Lawrence Sperry* (Charles Scribner, New York 1978)

29.8 J.F. Wagner: From Bohnenberger's machine to integrated navigation systems, 200 years of inertial navigation, Photogramm. Week (2005)

29.9 R.P. Feynman, R.B. Leighton, M. Sands: *The Feynman Lectures on Physics* (Addison-Wesley, Reading 1963)

29.10 T.F.T. Submarine: *Navpers 16160 Standards and Curriculum Division Training* (Bureau of Naval Personnel United States Navy, Arlington 1946)

29.11 G. Sagnac: L'ether lumineux demontre par l'effect du vent relatif d'ether dans un interferometre en rotation uniforme, C. R. Acad. Sci. Paris **157**, 708–710 (1913)

29.12 G. Sagnac: Sur la preuve de la realitet de l'ether lumineaux par l'experience de l'interferographe tournant, C. R. Acad. Sci. Paris **157**, 1410–1413 (1913)

29.13 F. Harress: Die Geschwindigkeit des Lichtes in bewegten Körpern, Ph.D. Thesis (Richters, Jena 1912)

29.14 A.A. Michelson, H.G. Gale: The effect of the Earth's rotation on the velocity of light, J. Astrophys. **61**, 140–145 (1925)

29.15 S. Ekeziel, H.J. Arditty: *Fiber-Optic Rotation Sensors*, Springer Ser. Opt. Sci. (Springer, Berlin, Heidelberg 1982)

29.16 G.E. Stedman: Ring-laser tests of fundamental physics and geophysics, Rep. Prog. Phys. **60**, 615–688 (1997)

29.17 D. Mackenzie: From the luminiferous ether to the Boeing 757: A history of the laser gyroscope, Technol. Cult. **34**(3), 475–515 (1993)

29.18 E. Foxlin, M. Harrington, Y. Altshuler: Miniature 6-DOF inertial system for tracking HMDs, SPIE Proc. 3362 (1998) pp. 214–228

29.19 M. Mostafa: History of inertial navigation systems in survey applications, J. Am. Soc. Photogramm. Remote Sens. **67**, 1225–1227 (2001)

29.20 A.J. Van Dierendonck, P. Fenton, T. Ford: Theory and performance of narrow correlator spacing in a GPS receiver, Navigation **39**, 265–283 (1992)

29.21 A. Rietdorf, C. Daub, P. Loef: Precise positioning in real-time using navigation satellites and telecommunication, Proc. 3rd Workshop Posit. Navig. Commun. (2006) pp. 123–128

29.22 J. Rios, E. White: Low cost solid state GPS/INS package, Proc. Inst. Navig. Conf. (2000)

29.23 R. Siegwart, I.R. Nourbakhsh: *Introduction to Autonomous Mobile Robots* (MIT Press, Cambridge 2004)

29.24 H.R. Everett: *Sensors for Mobile Robots: Theory and Application* (Peters, Wellesley 1995)

29.25 A. Leick: *GPS Satellite Surveying* (Wiley, Hoboken 2004)

29.26 P. Misra, P. Enge: *Global Positioning System: Signals, Measurements, and Performance* (Ganga-Jamuna Press, Lincoln 2006)

29.27 J. Farrell, M. Barth: *The Global Positioning System and Inertial Navigation* (McGraw-Hill, New York 1989)

29.28 M.S. Grewal, L.R. Weill, A.P. Andrews: *Global Positioning Systems, Inertial Navigation, and Integration* (Wiley, Hoboken 2007)

Avidbots Ex1016
Page 786 of 2154

Multimedia Contents

30. Sonar Sensing

Lindsay Kleeman, Roman Kuc

Sonar or ultrasonic sensing uses the propagation of acoustic energy at higher frequencies than normal hearing to extract information from the environment. This chapter presents the fundamentals and physics of sonar sensing for object localization, landmark measurement and classification in robotics applications. The source of sonar artifacts is explained and how they can be dealt with. Different ultrasonic transducer technologies are outlined with their main characteristics highlighted.

Sonar systems are described that range in sophistication from low-cost threshold-based ranging modules to multitransducer multipulse configurations with associated signal processing requirements capable of accurate range and bearing measurement, interference rejection, motion compensation, and target classification. Continuous-transmission frequency-modulated (CTFM) systems are introduced and their ability to improve target sensitivity in the presence of noise is discussed. Various sonar ring designs that provide rapid surrounding environmental coverage are described in conjunction with mapping results. Finally the chapter ends with a discussion of biomimetic sonar, which draws inspiration from animals such as bats and dolphins.

30.1 **Sonar Principles** 754
30.2 **Sonar Beam Pattern** 756
30.3 **Speed of Sound** 758
30.4 **Waveforms** 758
30.5 **Transducer Technologies** 759
30.5.1 Electrostatic 759
30.5.2 Piezoelectric 760
30.5.3 MEMS 760
30.6 **Reflecting Object Models** 760
30.7 **Artifacts** 761
30.8 **TOF Ranging** 762
30.9 **Echo Waveform Coding** 765
30.10 **Echo Waveform Processing** 767
30.10.1 Ranging and Wide-Bandwidth Pulses ... 767
30.10.2 Bearing Estimation 768
30.11 **CTFM Sonar** 769
30.11.1 CTFM Transmission Coding 769
30.11.2 CTFM TOF Estimation 769
30.11.3 CTFM Range Discrimination and Resolution 770
30.11.4 Comparison of CTFM and Pulse-Echo Sonar 770
30.11.5 Applications of CTFM 771
30.12 **Multipulse Sonar** 772
30.12.1 Interference Rejection 772
30.12.2 On-the-Fly Target Classification 772
30.13 **Sonar Rings and Arrays** 773
30.13.1 Simple Ranging Module Rings .. 773
30.13.2 Advanced Rings 773
30.13.3 Sonar Rings with FPGA Hardware Processing 774
30.13.4 Double Refresh Rate Sonar Ring 775
30.13.5 Sparse 3-D Arrays 775
30.14 **Motion Effects** 775
30.14.1 Moving Observation of a Plane . 776
30.14.2 Moving Observation of a Corner 776
30.14.3 Moving Observation of a Edge .. 777
30.14.4 The Effect of a Moving Observation on the Angle of Reception 777
30.14.5 Plane, Corner, and Edge Moving Observation Arrival Angles 777
30.15 **Biomimetic Sonars** 778
30.16 **Conclusions** 779
Video-References 780
References 780

Avidbots Ex1016
Page 787 of 2154

30.1 Sonar Principles

Sonar is a popular sensor in robotics that employs acoustic pulses and their echoes to measure the range to an object. Since the sound speed is usually known, the object range is proportional to the echo travel time.

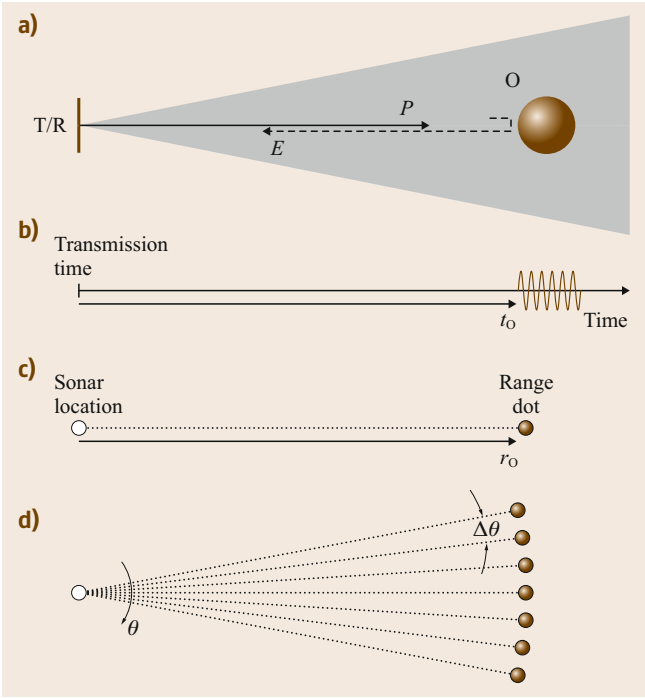


Fig.30.1a–d Sonar ranging principles: **(a)** sonar configuration, **(b)** echo waveform, **(c)** range dot placement, **(d)** sonar map

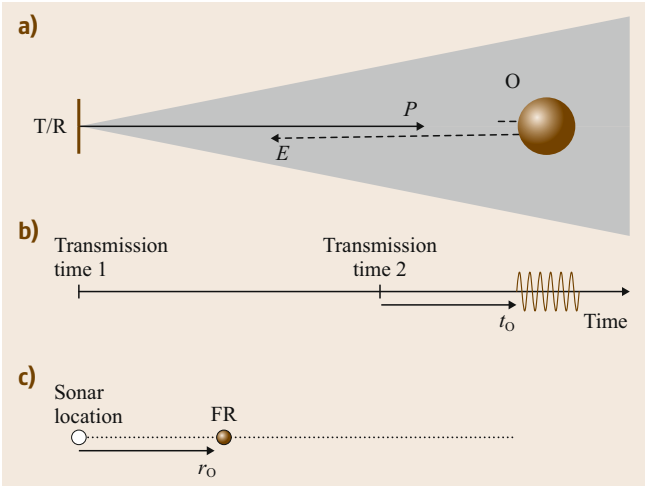


Fig.30.2a–c False range reading: **(a)** sonar configuration, **(b)** probing pulse 2 transmitted before echo from pulse 1 arrives, **(c)** false range (FR) is measured from transmission time 2

At ultrasonic frequencies the sonar energy is concentrated in a beam, providing directional information in addition to range. Its popularity is due to its low cost, light weight, low power consumption, and low computational effort, compared to other ranging sensors. In some applications, such as in underwater and low-visibility environments, sonar is often the only viable sensing modality.

Sonars in robotics have three different, but related, purposes:

1. Obstacle avoidance: the first detected echo is assumed to measure the range to the closest object. Robots use this information to plan paths around obstacles and to prevent collisions.
2. Sonar mapping: a collection of echoes acquired by performing a rotational scan or from a sonar array is used to construct a map of the environment. Similar to a radar display, a range dot is placed at the detected range along the probing pulse direction.
3. Object recognition: a sequence of echoes or sonar maps is processed to classify echo-producing structures composed of one or more physical objects. When successful, this information is useful for robot registration or landmark navigation.

Figure 30.1 shows a simplified sonar system, from its configuration to the resulting sonar map. A sonar transducer, T/R, acts as both the transmitter (T) of a probing acoustic pulse (P) and the receiver of echoes (E). An object O lying within the sonar beam, indicated by the shaded region, reflects the probing pulse. A part of the reflected signal impinges on the transducer as is detected as an echo. The echo travel time t_o, commonly called the *time-of-flight* (TOF) is measured from the probing pulse transmission time. In this case the echo waveform is a replica of the probing pulse, which usually consists of as many as 16 cycles at the resonant frequency of the transducer. The object range r_o is computed from t_O using

$$r_O = \frac{ct_O}{2}\,, \tag{30.1}$$

where c is the sound speed (343 m/s at standard temperature and pressure). The factor of 2 converts the round-trip ($P + E$) travel distance to a range measurement. Beam-spreading losses and acoustic absorption limit sonar range.

In forming a sonar map, a range dot is placed along the direction corresponding to the transducer's physical orientation. A sonar map is usually built by rotating the sensor about the vertical axis, indicated by the orientation angle θ, through a series of discrete angles

separated by $\Delta\theta$ and placing sonar dots the corresponding ranges. Since the range from the object O to the center of T/R is almost constant as T/R rotates, the range dots typically fall on a circle as long as O lies within the beam. Hence, sonar maps are made up of arcs.

The major limitations of sonar include:

1. The wide sonar beam causes poor directional resolution. Objects are located at the middle of isolated arcs, but shorter-range objects shorten the arcs of those at farther ranges, and the arcs produced by a collection of objects are often difficult to interpret. A consequence of this effect is that wide beams occlude small openings, limiting robot navigation.
2. The slow sound speed, relative to an optical sensor, reduces the sonar sensing rate. A new probing pulse should be transmitted after all detectable echoes from the previous pulse have expired, otherwise the false reading shown in Fig. 30.2 can occur. The echo from probing pulse 1 occurs after probing pulse 2 is emitted. Sonar measures the TOF from the most recent probing pulse. Many sonars transmit probing pulses every 50 ms, but encounter false readings in reverberant environments.
3. Smooth surfaces at oblique incidence do not produce detectable echoes. Figure 30.3 shows a planar surface (a wall) that acts as a mirror to the sonar beam. The important point is that the nearby wall does not itself produce a detectable echo, and a robot using sonar for obstacle avoidance may collide with the wall.
4. Artifacts caused by beam side-lobes and multiple reflections produce range readings in the environment where no objects exist. Figure 30.3 also shows the redirected beam enclosing object O. The echo is also redirected by the wall back to the transducer. From the transducer's reference, the object is at the virtual object location VO, and it would generate the same sonar map as shown in Fig. 30.1. Since there is no physical object corresponding to the sonar dot location, it is an artifact. Also, note that the acoustic energy indicated by the dot-dashed line reflected back to the transducer is not detected because it does not lie within the beam cone. Beam side-lobes often detect these echoes and produce shorter-range readings but placed along the sonar orientation.
5. Travel time and amplitude variations in the echoes caused by inhomogeneities in the sound speed. Both effects cause random fluctuations in the detected echo travel time, even in static environments. Figure 30.4 illustrates thermal fluctuations that cause speed up, retardation, and travel redirection by refraction of echoes. These cause temporal and amplitude variations in the echoes and jitter in the range readings. While these typically introduce minor changes in sonar maps, they often cause havoc with approaches using finer analysis.

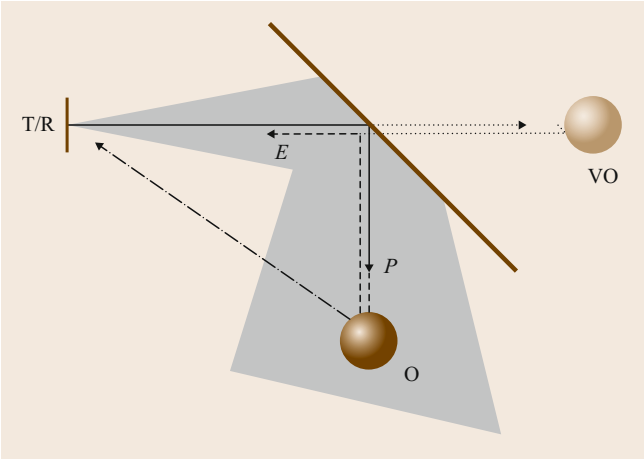


Fig. 30.3 The smooth surface redirects the beam, causing a sonar artifact at virtual object (VO) location. The *dot-dashed* echo path falls outside the sonar beam and does not produce a detectable echo

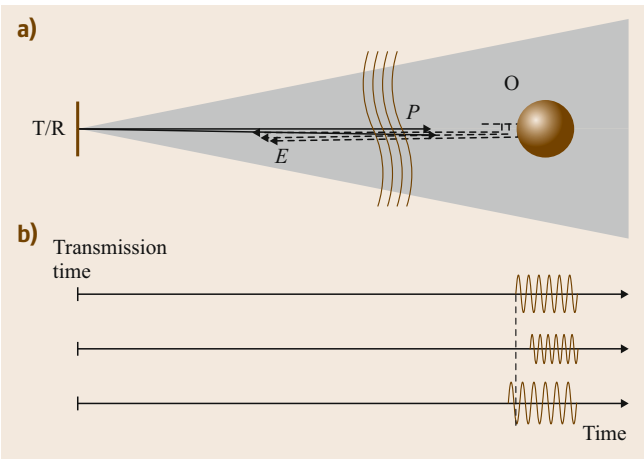


Fig.30.4a,b Random echo jitter. **(a)** Sonar configuration; thermal inhomogeneities in the acoustic transmission medium cause refraction effects. **(b)** Examples of variations in echo travel times and amplitudes in a static environment

This chapter describes the physical and mathematical details that extend this simplified sonar model to practical sonar systems.

30.2 Sonar Beam Pattern

To derive a qualitative description of the sonar transducer, we apply elementary acoustics theory to a simplified model to achieve a simple analytic form [30.1]. A sonar emitter is commonly modeled as a circular piston surface of radius a vibrating with frequency f in an infinite planar baffle. The wavelength λ is

$$\lambda = \frac{c}{f}\,, \tag{30.2}$$

where c the sound speed in air, 343 m/s at 25 °C [30.2]. When $a > \lambda$ the emitted pressure field forms a beam consisting of a main lobe surrounded by side-lobes. In the far field, or range greater than a^2/λ, the beam is described by its *directivity pattern*, which is the two-dimensional Fourier transform of the aperture function, in this case the circular aperture produces a Bessel function. The emitted pressure amplitude at range r and angle θ relative to the piston axis can be written as

$$P_{\mathrm{E}}(r,\theta) = \frac{\alpha a^2 f}{r}\left(\frac{2J_1(ka\sin\theta)}{ka\sin\theta}\right)\,, \tag{30.3}$$

where α is a proportionality constant that includes the density of air and the source strength, $k = 2\pi/\lambda$, and J_1 is the Bessel function of the first kind. The term in the brackets evaluates to 1 along the sonar axis, $\theta = 0$. The a^2 term indicates that the emitted pressure increases with the piston area. The frequency f appears in the numerator because the faster-moving piston generates higher pressures. The range r appears in the denominator because the conservation of energy requires the pressure to decrease as the beam widens with range.

The main lobe is defined by its first off-axis null, which occurs at an angle of

$$\theta_0 = \arcsin\left(\frac{0.61\lambda}{a}\right)\,. \tag{30.4}$$

For example, the popular electrostatic instrument grade transducer, formerly produced by Polaroid [30.3], had a radius of $a = 1.8$ cm and is conventionally driven at $f = 49.4$ kHz, making $\lambda = 0.7$ cm and $\theta_0 = 14.7°$.

An object that is small compared to λ and located in the emitted pressure field produces an echo with a spherical wavefront whose amplitude decays with the inverse of the distance propagated. In the common pulse-echo single transducer (*monostatic*) ranging sensor, only part of the echo wavefront impinges on the receiving aperture. The *sensitivity pattern* of the circular aperture, now acting as the receiver, has the same beam-like Bessel function form given in (30.3) by the reciprocity theorem [30.1]. If the reflecting object is located at (r,θ) relative to the transducer, the detected echo pressure amplitude, referenced to the receiver output, is given by

$$P_{\mathrm{D}}(r,\theta) = \frac{\beta f a^4}{r^2}\left(\frac{2J_1(ka\sin\theta)}{ka\sin\theta}\right)^2\,, \tag{30.5}$$

where β is a proportionality constant that includes parameters that cannot by controlled in a design, such as the density of air. The additional a^2 in the numerator occurs because larger apertures detect more of the echo wavefront.

Figure 30.5 shows the echo amplitude from a small (point-like) object located in the far field as a function

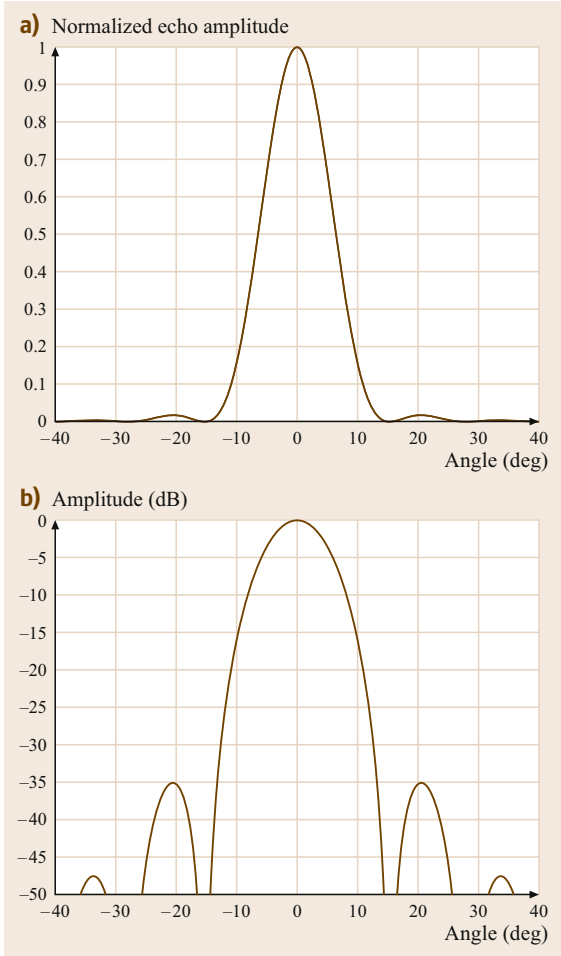


Fig.30.5a,b Normalized amplitude of echo from small object predicted by the piston model as a function of angle. **(a)** Linear scale. **(b)** Decibel scale

of angle detected by the electrostatic instrument grade transducer. The curve has been normalized by the on-axis echo amplitude.

This model is qualitative in that it provides the following practically useful insights:

- For a small reflector size relative to the wavelength, the echo amplitude decreases inversely with the square of the range because there is a $1/r$ dispersion loss from the transmitter to the object, followed by an additional $1/r$ dispersion loss in the echo back to the receiver. However, larger reflectors can be treated using a Huygens principle approach [30.4] by dividing them into smaller reflectors and coherently adding their echo contributions. When this is done in two dimensions over a normally incident extended plane reflector, the echo amplitude decreases as $1/r$ rather than $1/r^2$. A cylindrical reflector extends over one dimension and results in an amplitude variation with range between $1/r$ and $1/r^2$. A more-extreme situation can occur with a concave reflector that acts as an acoustic magnifier, resulting in an amplitude that decreases with a negative power of range of less than one.
- The transducer excited by an approximation to a sinusoid exhibits side-lobes due to null caused by phase cancellation. For example, the 16-cycle excitation employed in the conventional sonar exhibits side-lobes. The peak of the first side-lobe is −35 dB relative to the echo amplitude when a small reflector lies on the transducer axis. The specification sheet for the 600 series instrument-grade transducer shows the first off-axis null at 15° and a first side-lobe peak magnitude of −26 dB. We presume that these measurements were made using a plane as a reflector.
- This model can be used to compute approximate beam parameter values for other common transducers. For example, the SensComp 7000 Series [30.5] with $a = 1.25$ cm yields $\theta = 20°$, equal to the specified value. However, the specified first side-lobe peak magnitude equals approximately −16 dB, which is substantially different from the expected −35 dB.

The limitations of the qualitative model include:

- Actual transducers only approximate pistons vibrating in an infinite planar baffle. The infinite baffle directs all the radiated sound pressure into the half-space in front of the transducer. Actual transducers radiate in all directions, but most of the acoustic energy is concentrated within the main lobe.
- All pulse-echo ranging sonars operate with finite-duration pulses rather than infinite-duration sinusoids. Several systems described below use pulses that are quite different from a sinusoidal excitation, either in duration or in form. These are commonly analyzed by computing the spectrum of the pulse and decomposing it into several sinusoidal frequencies, each having its own beam pattern. For example, the echo amplitude predictions above are reasonably accurate, including beam width and side-lobes, for the 16-cycle pulses. However, when impulse or swept-frequency excitations are used, the net beam profile becomes the superposition (of linear amplitudes) of the beam patterns produced by each frequency component in the excitation. Such broadband excitations do not exhibit nulls because the nulls formed by one frequency are filled in by main and side-lobes of beams produced by other frequencies.
- Most sonar transducers are encased in protective housings. The electrostatic instrument-grade transducer cover forms a mechanical filter that enhances the acoustic output at 49.4 kHz. The cases of other transducers may distort the transmitted field, but most form some type of directional beam.
- The model does not include frequency-dependent acoustic absorption of the transmission medium. These reduce the echo amplitudes predicted by the model.

The analytic model above is limited to simple configurations. With current computational power, transducers can be extended to those with arbitrary, even multiple, apertures and with various excitations. Waveforms of echoes from objects having arbitrary shapes can be simulated by using Huygens principle [30.4]. The transmitter, receiver, and object surfaces are broken up into two-dimensional (2-D) surface arrays of emitting, reflecting, and detecting elements, using squares of dimension $< \lambda/5$ (the smaller the better, but taking longer). The impulse response of a given configuration is computed by assuming an impulsive emission and superimposing the travel times along all possible paths from all transmitter elements to all object elements and then to all receiver elements. The temporal resolution should be $< (20 f_{\max})^{-1}$, where $f_{\max}$ is the maximum frequency in the excitation. A 1 —s resolution is adequate for a 16-cycle 49.4 kHz excitation. A much finer resolution (< 0.1 —s) is required for an impulsive excitation. The echo waveform is then computed as the convolution of this impulse response with the actual transmitted pulse waveform [30.4].

Avidbots Ex1016
Page 791 of 2154

30.3 Speed of Sound

The speed of sound c varies significantly with atmospheric temperature, pressure, and humidity and can be critical in determining the accuracy of a sonar system. This section outlines the relationship between c and these variables and is based on [30.6, 7].

The speed of sound in dry air at sea-level air density and one atmosphere pressure is given by

$$c_T = 20.05\sqrt{T_C + 273.16}\,\mathrm{ms}^{-1}\,, \tag{30.6}$$

where T_C is the temperature in degrees Celsius. Under most conditions (30.6) is accurate to within 1%. However, should the relative humidity be known, a better estimate can be made as

$$c_H = c_T + h_r\left[1.0059\times10^{-3} + 1.7776\times10^{-7}(T_C + 17.78)^3\right]\mathrm{ms}^{-1}\,. \tag{30.7}$$

Equation (30.7) is accurate to within 0.1% for temperatures in the range −30 to 43 °C for most pressures at sea level. Should atmospheric pressure, p_s be known then the following expression can be used

$$c_P = 20.05\sqrt{\frac{T_C + 273.16}{1 - 3.79\times10^{-3}\left(\frac{h_r p_{sat}}{p_s}\right)}}\,\mathrm{ms}^{-1}\,, \tag{30.8}$$

where the saturation pressure of air, p_{sat} is dependent on temperature as follows

$$\begin{aligned}\log_{10}\left(\frac{p_{sat}}{p_{s0}}\right) &= 10.796\left[1 - \left(\frac{T_{01}}{T}\right)\right] - 5.0281\log_{10}\left(\frac{T_{01}}{T}\right) \\ &+ 1.5047\times10^{-4}\left(1 - 10^{-8.2927\left[\left(\frac{T}{T_{01}}\right)-1\right]}\right) \\ &+ 0.42873\times10^{-3}\left(-1 + 10^{4.7696\left[1-\left(\frac{T_{01}}{T}\right)\right]}\right) \\ &- 2.2196\,,\end{aligned} \tag{30.9}$$

p_{s0} is the reference atmospheric pressure of 101.325 kPa and T_{01} is the triple-point isotherm temperature with the exact value of 273.16 K.

30.4 Waveforms

Sonars employ a variety of waveforms, the most common types of which are shown in Fig. 30.6. Each waveform can be considered the echo from a normally incident plane. Waveforms are classified as being narrow- or wide-band depending on their spectral bandwidth. Narrow-band pulses provide superior detection performance in the presence of additive noise, while wide-band pulses provide better range resolution and do not have side-lobes.

Figure 30.6a shows the waveform produced by the Murata 40 kHz piezoelectric transducer excited by an eight-cycle 40 kHz square wave 40 V_{rms}. The Murata sensor is small, lightweight, and efficient, but has an approximately 90° beam width. These transducers are used in monostatic, bistatic, and multiple transducer arrays [30.8, 9].

The next three waveforms were produced by the Polaroid 600 electrostatic transducer. Similar waveforms are generated by the smaller Polaroid 7000 transducer. Figure 30.6b shows the waveform produced by the 6500 ranging module. This ranging module with its 10 m range, low cost, and simple digital interface is a popular choice for implementing sonar arrays and rings. While the electrostatic transducer is inherently wide-band, with a usable frequency range from 10 to 120 kHz [30.10], narrow-band pulses are produced by exciting the transducer with 16 cycles at 49.4 kHz. Figure 30.6c illustrates a means to exploit the wide bandwidth of the Polaroid electrostatic transducer by exciting it with a decreasing-frequency square wave. Such frequency-sweep pulses are processed by a band of band-pass filters to extract the frequency dependence of reflecting objects. A correlation detector, also know as a matched filter, compresses swept-frequency pulses to improve range resolution. Longer-duration (100 ms) pulses are used in CTFM systems. Figure 30.6d shows a wide-band pulse when the excitation is a 10 µs-duration 300 V pulse. The metal protective mesh, which also acts as a mechanical filter resonant at 50 kHz, was removed by machining to achieve a usable bandwidth from 10 to 120 kHz, with the peak occurring at 60 kHz. Such wide-band pulses are useful for object classification [30.10, 11]. These pulses have small amplitudes, limiting their range to 1 m or less.

Avidbots Ex1016
Page 792 of 2154

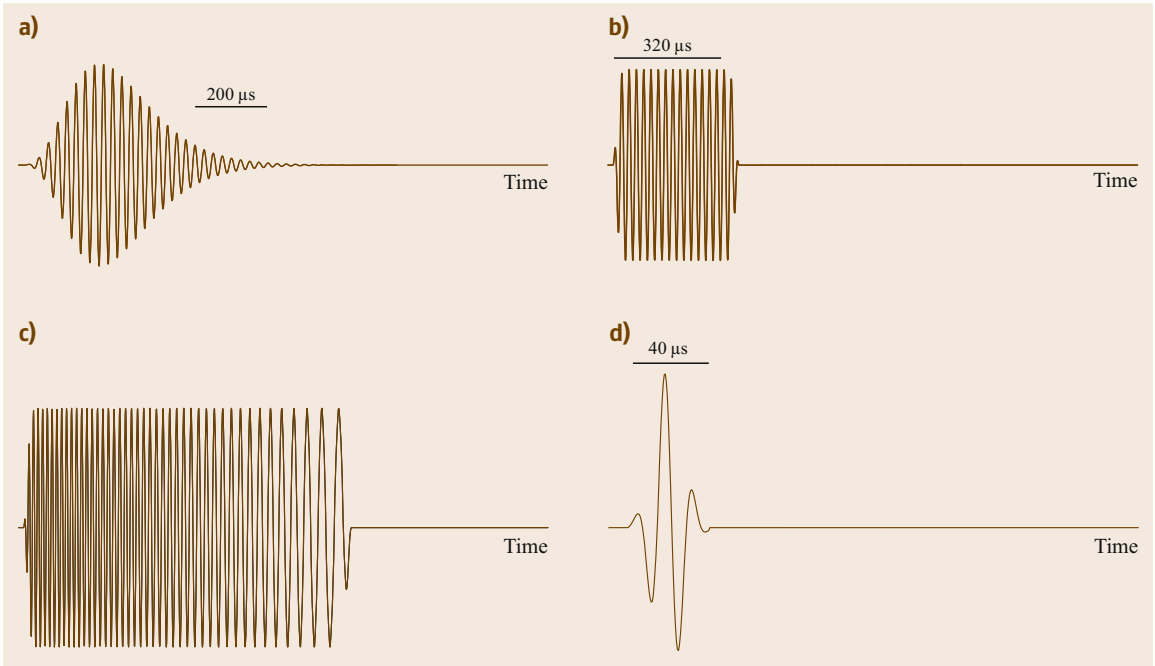


Fig.30.6a–d Common sonar pulse waveforms. **(a)** Murata 40 kHz transducer (narrow band). **(b)** Polaroid 600 electrostatic transducer excited with 16-cycle sinusoid at 49.4 kHz (6500 ranging module – narrow band). **(c)** Polaroid 600 electrostatic transducer excited with decreasing-frequency excitation signal (wide band). **(d)** Polaroid 600 electrostatic transducer excited with 10 µs 300 V pulse (wide band)

30.5 Transducer Technologies

Electrostatic and piezoelectric transducers are the two major types available that operate in air and can in principle operate both as a transmitter and receiver – some samples are shown in Fig. 30.7. In general electrostatic devices have a higher sensitivity and bandwidth but typically require a bias voltage above 100 V. Piezoelectric devices operate at lower voltages, making their electronic interfacing simpler, but have a high-Q resonant ceramic crystal and this results in a narrow frequency response compared to electrostatic transducers.

30.5.1 Electrostatic

An example of an electrostatic transducer is the Polaroid instrument-grade transducer (now available from SensComp.com) constructed from a gold-coated plastic foil membrane stretched across a round grooved aluminium back plate. The conductive foil is charged via a bias voltage of 150 V with respect to the back plate. Incoming sound waves vibrate the foil and change the average distance between the foil and back plate and thereby changing the capacitance of the foil. Assuming that the charge q is constant, the voltage $v(t)$ is generated proportional to this varying capacitance $C(t)$ as $v(t) = qC(t)$. As a transmitter, the transducer membrane is vibrated by applying 0–300 V pulses across this capacitor, typically using a pulse transformer. The



Fig. 30.7 *Left to right*: Series 9000, instrument-grade, and Series 7000 transducers; front and back views are shown (courtesy Acroname, Inc., Boulder)

Avidbots Ex1016
Page 793 of 2154

charge induced by the 300 V on the capacitor causes an electrostatic attraction force between the membrane and the back plate. The grooves on the back plate allow stretching of the membrane and by creating randomness in the back plate roughness a broad resonance can be achieved in the frequency response. For example the bandwidth of the 7000 series Polaroid transducer is 20 kHz. A front grille is mounted on the transducer and removing this grille reduces losses and reverberation between the grill and the membrane. Another electrostatic transducer was designed by Kay and details of its design can be found in [30.12].

30.5.2 Piezoelectric

Piezoelectric ceramic transducers can be used as both transmitters and receivers, however some manufacturers sell transmitters and receivers separately in order to optimize the transmitted power and receiver sensitivity, respectively. A piezoelectric resonant crystal mechanically vibrates when a voltage is applied across the crystal, and in reverse generates a voltage when mechanically vibrated. Often a conical concave horn is mounted on the crystal to acoustically match the crystal acoustic impedance to that of air. An example is the Murata MA40A5R/S receiver and sender transducers, which operate at 40 kHz. This device has a diameter of 16 mm and a 60° beam angle for transmitter combined with receiver for −20 dB loss compared to the maximum sensitivity. The effective bandwidth of transmitter and receiver is only a few kHz due to the resonant nature of the crystals. This limits the envelope rise time of pulses to around 0.5 ms. An advantage is the ability to drive piezoelectric devices with low voltages, for example by connecting each terminal to complementary CMOS logic outputs. There is a wide range of resonant frequencies for piezoelectric transducers from 20 kHz to megahertz. Also available is piezoelectric film called polarized fluoropolymer, polyvinylidene fluoride (PVDF) from [30.13]. This flexible film can be cut to shape and custom ultrasonic transmitters and receivers can be formed. The sensitivities of the transmitters and receivers made from PVDF is generally lower than that of ceramic crystal transducers and most applications are short range where the broadband nature of PVDF allows short pulses to be formed, allowing pulse-echo ranging to as little as 30 mm.

30.5.3 MEMS

Microelectromechanical system (MEMS) ultrasonic transducers can be fabricated on a silicon chip and integrated with electronics. The sensors offer a low-cost mass-produced alternative to standard transducers. MEMS ultrasonic transducers operate as electrostatic capacitive transducers where the membrane can be made from thin nitride. Devices operate at frequencies up to several megahertz and offer advantages in signal-to-noise ratio over piezoelectric devices due to their better matching to air acoustic impedance [30.14]. Two-dimensional arrays of devices can be deployed on a chip that are well matched and steerable.

30.6 Reflecting Object Models

Modeling the reflection processes helps in interpreting echo information. In this section we consider three simple reflector models: planes, corners, and edges, shown in Fig. 30.8. These models apply to both single transducers and arrays.

A plane is a smooth surface that acts as an acoustic mirror. Smooth walls and door surfaces act as planar reflectors. The plane must be sufficiently wide to produce the two reflections whose path is shown in dotted line. The plane reflector is then slightly larger than the intersection area of the beam with a plane of infinite extent. Smaller planes produce weaker echoes because of a smaller reflecting surface and negative interference by echoes diffracted from the edges of the plane. An acoustic mirror allows the analysis using a virtual transducer, indicated by primes in the figure.

A corner is the concave right-angle intersection of two surfaces. Corners formed by intersecting walls, the sides of file cabinets, and door jambs are commonly observed corner reflectors in indoor environments. The novel feature of the corner, and its three-dimensional (3-D) counterpart the corner cube, is that waves reflect back in the same direction from which they originate. This is caused by planar reflections at each of the two surfaces defining the corner. The virtual transducer is then obtained by reflecting the transducer about one plane of the corner and then the other plane. This gives rise to a reflection through the intersection point of the corner as shown in Fig. 30.8b. The virtual transducer analysis indicates that, for a monostatic sonar, echoes from a plane and corner are identical and that planes and corners can generate identical sonar maps [30.4]. The difference in the virtual transducer orientation between planes and corners has been exploited using transducer arrays to differentiate these reflectors [30.11, 15].

Avidbots Ex1016
Page 794 of 2154

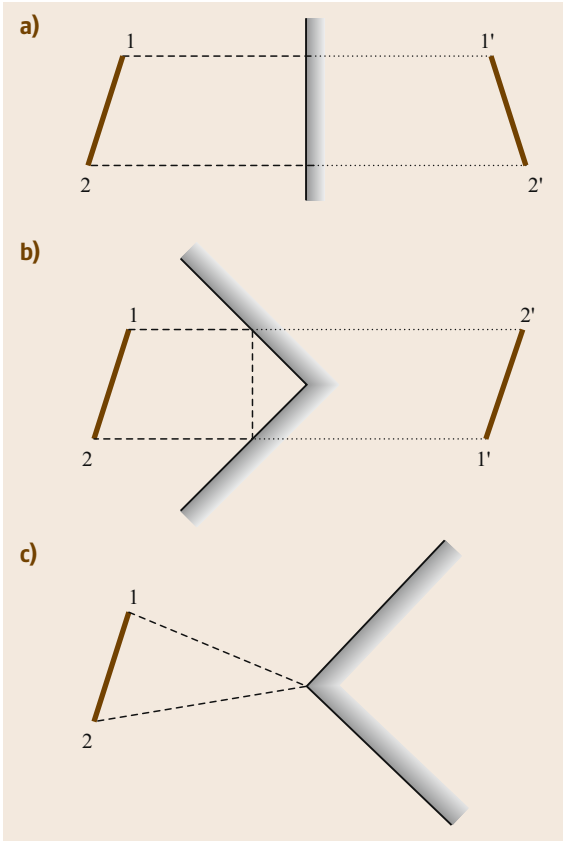


Fig.30.8a–c Reflector models: **(a)** plane, **(b)** corner, and **(c)** edge

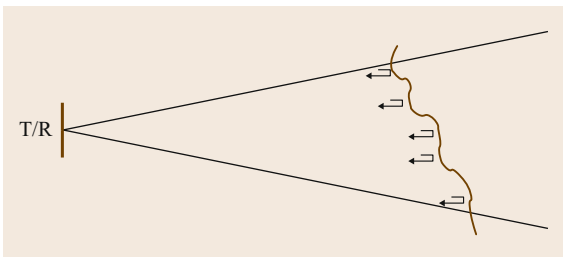


Fig. 30.9 Random reflector model. Echoes reflect back from normally incident surface sections within the beam

The edge shown in Fig. 30.8c models physical objects such as convex corners and high-curvature surfaces (posts), where the point of reflection is approximately independent of transducer position. Edges are encountered in hallways. While planes and corners generate strong echoes, edges generate weak echoes that are detected only a short range [30.4], making them difficult objects to detect. Early robot sonar researchers placed bubble wrap material on edge surfaces to make them reliably detectable.

Many environmental objects can be configured as a collection of planes, corners, and edges. Models for echo production [30.16, 17] indicate that normally incident surface patches and locations at which sharp changes in the surface function and its derivatives generate echoes. Objects with rough surfaces or a collection of many objects generate echoes from a variety of ranges and bearings, as illustrated in Fig. 30.9. If $p(t)$ represents a single echo waveform, often a replica of the probing waveform, the total echo waveform $p_T(t)$ is the sum of individual echoes $p_i(t)$ from N normally incident patches at range r_i and bearing θ_i, scaled by amplitude a_i, or

$$p_T(t) = \sum_{i=1}^{N} a_i(\theta_i) p_i\left(t - \frac{2r_i}{c}\right), \qquad (30.10)$$

where $a_i(\theta_i)$ is an amplitude factor related to the surface patch size and its bearing in the beam. Wide-bandwidth echoes are more complicated because their waveform changes in a deterministic fashion due to diffraction [30.11].

Sonars that analyze $p_T(t)$ employ analog-to-digital converters to obtain waveform samples [30.11, 18]. Reflecting patches separated in range produce isolated patches [30.11], but more often the incremental travel time is less than the pulse duration, causing pulse overlap. Rough surfaces and volume scatterers, such as indoor foliage, have large N, allowing $p_T(t)$ to be treated as a random process [30.19, 20]. Conventional TOF sonars output the first time that $p_T(t)$ exceeds a threshold [30.11].

30.7 Artifacts

Sonars usually work well in simple environments, while complex environments often produce mysterious readings, artifacts, that foil attempts to build reliable sonar maps. Artifacts have given sonar a bad reputation as being a noisy, or low-quality, sensing modality. Sonar stalwarts believe sonar would open up many new applications, *if only* we understood echoes at a level that approximates that employed by bats and dolphins [30.21]. Sonar stalwarts divide into two categories in terms of how they treat artifacts. The first attempts to build *intelligent sensors* that identify and suppresses artifacts before transmitting data to a higher-level reasoning pro-

Avidbots Ex1016
Page 795 of 2154

gram. Previous approaches [30.22, 23] required custom electronics, which other researchers have been reluctant to adopt because of expense or lack of experience. An alternate approach is to control conventional sonar in a novel fashion to produce a series of spikes and requires only software changes [30.24]. Sonar arrays have been used to find consistent data [30.25–27]. Echoes from specular reflectors, such as planes, corners, or posts, exhibit detectable features, which can be obscured by artifacts.

The second category of sonar users attempts to eliminate artifacts produced by conventional sensors by using higher-level postprocessing. These include proponents of occupancy (or certainty) grids [30.28, 29], including those that apply simplified physical models, such as sonar arcs [30.30, 31]. In simple environments postprocessing usually eliminates artifacts that are inconsistent with a feature [30.32] or with a physical map [30.31]. More-sophisticated methods handle artifacts by treating them as noise and applying hidden Markov models (HMM) [30.33]. However, multiple passes are needed to successfully teach the system about relatively simple environments, mostly because artifacts are not amenable to being treated as independent additive noise. Eliminating troublesome artifacts would replace HMM with simpler Markov chains [30.34, 35], and sufficient sonar data can be obtained in a single pass. What frustrates this second category, and mildly amuses the first, is that this postprocessing works well *in simple environments*, but fails in real-world environments. This second category eventually abandons sonar and joins the camera and laser ranging crowd.

There are two important classes of artifacts: axial multiple reflection (MR) artifacts and dynamic artifacts. These artifacts are important in sonar mapping when they indicate the presence of a static object at a location where none exists. Troublesome MR artifacts are caused by delayed echoes produced by a previous probing pulse exceeding the detection threshold after the current probing pulse has been transmitted. Such artifacts then appear as close-range objects and obscure actual farther-range objects in conventional sonars. Most sonars employ probing pulse emission periods longer than 50 ms to avoid MR artifacts, although some reverberant environments can still produce artifacts [30.36].

Dynamic artifacts are produced by moving objects, such as individuals passing through the sonar beam. Even though these are actual objects and echoes indicate their true range, their presence should not be part of a sonar map that describes the static environment. Such dynamic artifacts make quantitative matchings between stored and generated sonar maps error-prone.

Another common artifact is a nonaxial MR artifact [30.4] caused by an obliquely incident smooth surface that redirects the sonar beam to some other echo-producing object. The TOF produces a range reading that is positioned along the sonar axis. While the object is not at the location indicated on a sonar map, its location in the sonar map is a stable element and can be useful for navigation.

One may argue that, if the locations of all objects are known, the echoes can be determined and should not be treated as random processes. However, the presence of speed fluctuations in the medium due to thermal gradients and ever-present electronic noise cause random fluctuations in the times that thresholds are exceeded. Even a stationary sonar in a static environment exhibits random fluctuations [30.37], similar to the visual experience of fading when viewing objects beyond a heated surface.

Sonar can identify artifacts by applying three physical criteria that are met by echoes from static environmental objects. Artifact features include [30.36]:

1. Echo amplitude – echoes with amplitudes less than a specified threshold
2. Coherence – echoes forming constant-range azimuthal intervals less than a specified threshold, and
3. Coincidence – echoes detected with a sonar array at different times (lacking temporal coherence) or corresponding to different locations (lacking spatial coherence).

30.8 TOF Ranging

Most conventional sonars employ Polaroid 6500 ranging modules [30.38] connected to the Polaroid 600 series electrostatic ultrasound transducer. The module is controlled with digital signals on two input lines (*INIT* for initialization and probing pulse transmission and *BLNk* for clearing the indication and resetting the detector) and the TOF reading occurs on its output line (*ECHO*). A logic transition on *INIT* causes the transducer to emit a pulse lasting for 16 cycles at 49.4 kHz. The same transducer detects echoes after a short delay to allow transmission transients to decay. Another interrogation pulse is typically emitted only after all the echoes produced by the previous pulse have decayed below a detection threshold.

Avidbots Ex1016
Page 796 of 2154

The module processes echoes by performing rectification and lossy integration. Figure 30.10 illustrates a simulation of the processed waveform applied to the threshold detector. While the echo arrives at time t_0 after the emission, *ECHO* exhibits a transition at the measured TOF time t_m, the first time the processed echo signal exceeds a detection threshold τ. By convention, the range r of the reflecting object is calculated by

$$r = \frac{ct_m}{2}\,, \tag{30.11}$$

where c is the speed of sound in air, usually taken as 343 m/s.

Figure 30.10b shows details around the threshold detection point including the residual high-frequency ripple after full-wave rectification and integration. Two effects can be noticed. First, t_m will always occur after t_0, making threshold detection a biased estimate of the true echo arrival time. Moreover, this bias is related to the echo amplitude: stronger echoes will produce an integrator output having a greater slope, which exceeds τ sooner than t_m. Second, as the echo amplitude decreases, for example, when the object moves away from the transducer axis, the threshold level occurs later in the integrator output and t_m will experience small jumps in time approximately equal to half the period [30.39].

The first step in developing a model of the detection process is to develop a model for the echo amplitude as a function of bearing. The Polaroid transducer is often modeled as a vibrating piston to yield the transmitter/receiver beam pattern shown in Fig. 30.11. To simplify the analysis, the peak of the beam profile is approximated with a Gaussian function, a parabola in logarithmic units in Fig. 30.11, to determine the echo amplitude as a function of object bearing θ, or

$$A_\theta = A_0 \exp\left(-\frac{\theta^2}{2\sigma^2}\right), \tag{30.12}$$

where A_0 is the on-axis amplitude and σ is a measure of the beam width. The value $\sigma = 5.25°$ provides a good fit around the peak of the beam pattern. The Gaussian model is reasonable only over the central section of the main lobe that produces detectable echoes.

We assume that the echo arrival time t_0 does not change significantly with transducer orientation (object bearing); this effect was investigated and found to be minor [30.4]. In contrast, measured TOFs, denoted by t_m and t'_m in Fig. 30.12, are amplitude dependent and a function of object bearing, which affects echo amplitude as shown in Fig. 30.11.

The module processes the detected echo waveform by rectification and lossy integration, as discussed above. To derive a useful analytic model, assume that the integration is lossless and the rectified echo is a unit step function with amplitude A. This approximates the processed waveform shown in Fig. 30.10b by a linear function around time t_m, shown in Fig. 30.12. The model ignores the residual ripple and the decreasing slope of the waveform as the lossy rectification approaches a constant value, shown in Fig. 30.10. The linear function with a slope proportional to the echo amplitude is given by $A_\theta(t - t_0)$, for $t \geq t_0$. This function

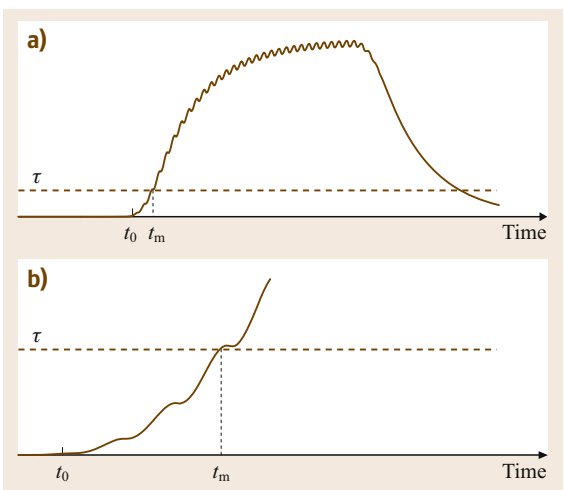


Fig.30.10a,b Simulation of Polaroid ranging module operation. **(a)** Processed echo waveform. **(b)** Expanded time and amplitude scale around threshold crossing point

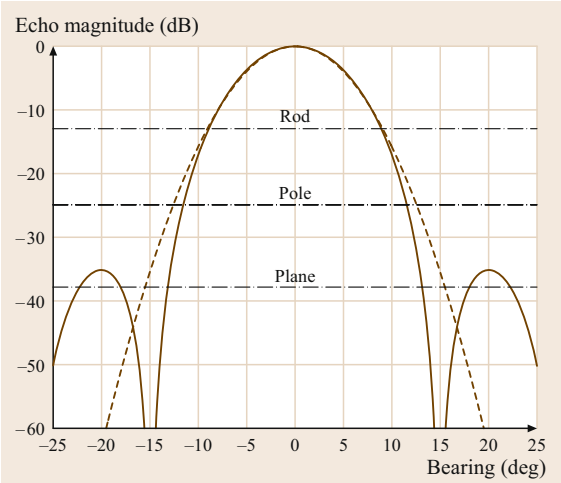


Fig. 30.11 Piston model transmitter/receiver pattern for the Polaroid 600 series transducer. A Gaussian approximation with SD = 5.25° is shown in *dashed line*. Equivalent threshold levels for plane, pole, and rod objects at 1.5 m range are shown as *dot-dashed lines*

Avidbots Ex1016
Page 797 of 2154

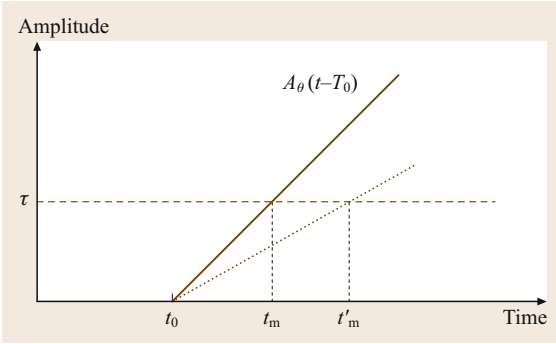


Fig. 30.12 TOF values t_m and t'_m for idealized processed echo waveforms having two amplitudes. The *solid line* indicates a larger-amplitude echo

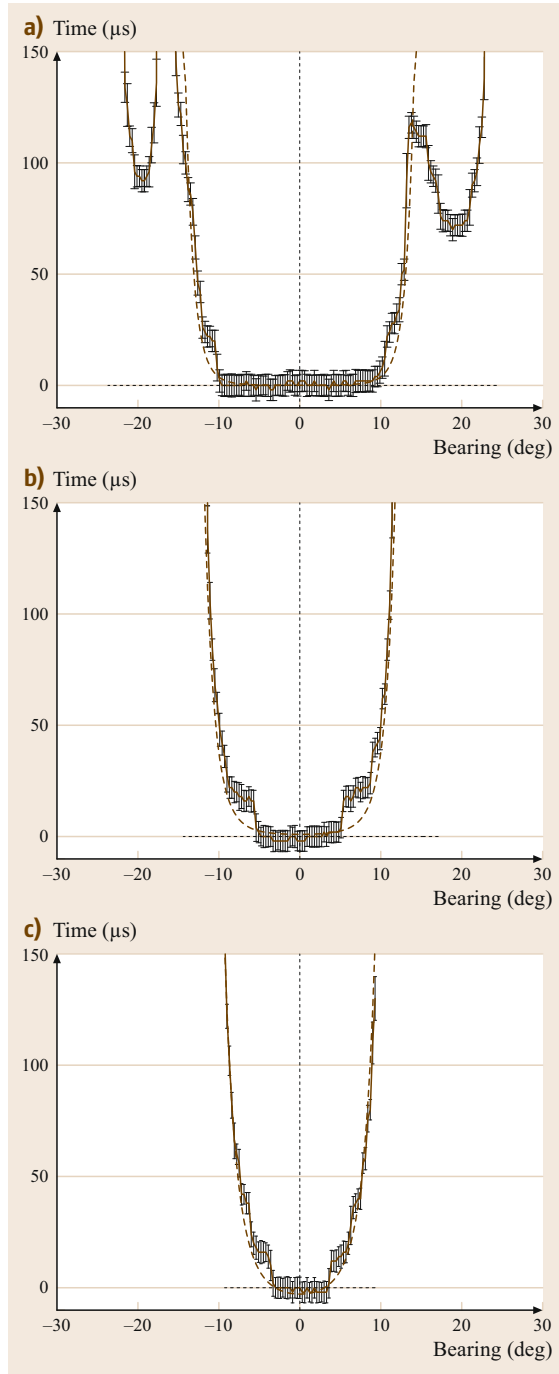


Fig.30.13a–c TOF data from an object at 1.5 m range; mean of 100 measurements with bars indicating ±1 SD. The *dashed lines* are model predictions. **(a)** 1 m wide plane ($\tau/A = 0.15\,\mu s$). **(b)** 8.9 cm diam pole ($\tau/A = 0.67\,\mu s$). **(c)** 8 mm diam rod ($\tau/A = 2.68\,\mu s$) ►

exceeds the threshold τ at

$$t_m = t_0 + \frac{\tau}{A_\theta} = t_0 + \frac{\tau}{A_0}\exp\left(\frac{\theta^2}{2\sigma^2}\right). \qquad (30.13)$$

For fixed τ, the incremental delay in t_m is a function of the bearing θ and inversely proportional to the echo amplitude. When a constant echo amplitude A (in volts) is applied to the integrator, the slope of the linear output is A V/s, with typical values on the order of $A_\theta = 10^5$ V/s. If $\tau = 0.10$ V, $\tau/A_\theta = 10^{-6}$ s $= 1\,\mu s$.

Experiments were conducted with a Polaroid 600 series transducer connected to a model 6500 ranging module [30.39]. The Polaroid module was operated conventionally to generate t_m values as a rotational scan was performed. Objects include a 1 m-wide plane, a 8.9 cm-diameter pole, and an 8 mm-diameter rod, all located at 1.5 m range. A rotational scan was performed from −40° to +40° in 0.3° steps. At each angle, 100 t_m values were recorded. The mean deviations from the t_m when the object is on the sonar axis ($\theta = 0$) were determined and the standard deviation (SD) values were computed. There were no other objects in proximity to the object being scanned. Echoes from objects beyond 2 m were eliminated by a range gate.

Figure 30.13a shows the data for the plane, Fig. 30.13b for the pole, and Fig. 30.13c for the rod. The values are shown relative to the t_m value observed at 0° bearing. Dashed lines indicate the values predicted by the model. The t_m values showed a variation with SD = 5μs (0.9mm) at zero bearing, which is about nine times greater than that predicted by sampling jitter alone. This random time jitter is caused by dynamic thermal inhomogeneities in the air transmission medium, which change the local sound speed and cause refraction [30.37, 40]. The SD increases with deviation from zero bearing because smaller echoes exceed the threshold later in the processed waveform. The smaller

slope of the latter part of the processed echo waveform shown in Fig. 30.10 causes greater t_m differences for a given variation in echo amplitude, thus increasing the SD.

One feature in the data not described by the model is due to residual ripple in the integrator output, which causes jumps in TOF readings equal to half periods (10μs) added to the value predicted by (30.13). These jumps are clearly evident in the mean values of Fig. 30.13.

The angular extent over which echoes were detected was 45° for the plane, 22.8° for the pole, and 18.6° for the rod. Side-lobes produced by the plane are visible and have small echo amplitudes, which cause their t_m values to be retarded in time. These angular extents can be related to the echo amplitudes that would have produced the respective arcs according to the piston model; these are indicated in Fig. 30.11. For the plane, the threshold level relative to the maximum echo amplitude is −38 dB, for the pole −25 dB, and for the rod −13 dB. Since the ranging module threshold at 1.5 m range is the same for each object, the difference in levels indicates the relative echo strength from each object, i. e., the plane echo is 13 dB (a factor of 4.5) greater than the pole echo, and the pole echo is 12 dB (a factor of 4) greater than the rod echo.

30.9 Echo Waveform Coding

Systems that display echo information beyond the first echo have been investigated [30.11, 18, 25, 41–43], but typically employ custom electronics. One motivation for examining the entire echo waveform is the success of diagnostic medical ultrasound imaging systems, which adopt this method [30.44, 45].

As a less expensive alternative to analog-to-digital conversion, the Polaroid ranging module can detect echoes beyond the initial echo by repeatedly resetting the detection circuit. The 6500 module specification suggests a delay before resetting to prevent the current echo from retriggering the detection circuit [30.3]. Let us ignore this suggestion and control the Polaroid module in nonstandard way to provide information about the entire echo waveform. Since the echo amplitude is estimated from the digital output produced by the Polaroid module, this operation has been called *pseudo-amplitude scan* (PAS) sonar [30.24].

The conventional ranging module processes detected echoes by performing rectification and forming a lossy integration, as illustrated in Fig. 30.14a.

The *BLNK* input is typically kept at zero logic level, which enables the *ECHO* output. *ECHO* exhibits a transition at the time when the processed echo signal exceeds a threshold, as shown in Fig. 30.14b. By convention, the time interval between the *INIT* and *ECHO* transitions indicates the time of flight (TOF), from which the range r of the reflecting object is calculated by

$$r = \frac{c \times \mathrm{TOF}}{2} . \qquad (30.14)$$

Echoes occurring after the initial echo can be detected by resetting *ECHO* by pulsing the *BLNK* input. The specification suggests that the *BLNK* pulse should be delayed after the *ECHO* indication by at least 440 μs to account for all 16 returning cycles in the echo and to allow it to decay below the threshold for the largest observable echo. The largest echoes typically saturate the detection circuit, providing a predetermined maximum value. This duration corresponds to the time interval over which the processed signal is above the threshold, as shown in Fig. 30.14a.

When an *ECHO* event is observed, the PAS system issues a short 3 μs (corresponding to a software query period) pulse on the *BLNK* input line, which clears the *ECHO* signal as shown in Fig. 30.14c. Upon being cleared, the Polaroid module exhibits a delay inversely related to the echo amplitude, lasting at least 140 μs for large-amplitude echoes, and then produces

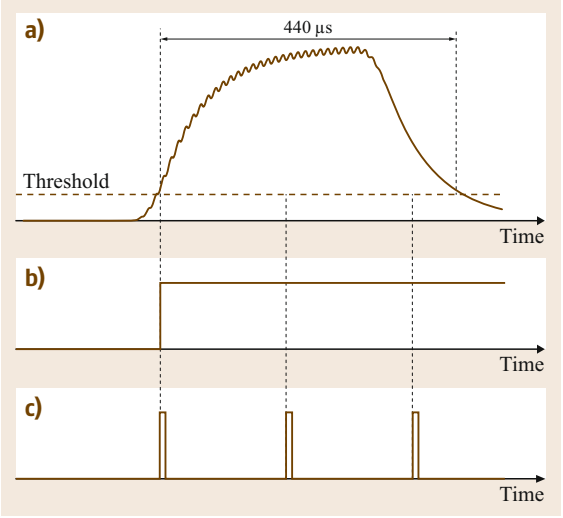


Fig.30.14a–c Polaroid ranging module operation modes. **(a)** Processed echo waveform. **(b)** *ECHO* output produced in conventional time-of-flight mode. **(c)** *ECHO* output produced in PAS mode

another *ECHO* event if the processed echo signal still exceeds the threshold. The PAS system repeatedly issues a *BLNK* pulse whenever an *ECHO* event is observed. Hence, a strong echo is represented by three pulses on the *ECHO* line, the first corresponding to the conventional TOF, followed by two more pulses. Because lower amplitude echoes spend less time above the threshold, a weaker echo produces two pulses spaced farther apart, and a very weak echo may produce only one pulse. Shorter *INIT* pulses discharge the integrator by a lesser amount and thus increase the number of pulses produced by the standard Polaroid emission [30.46, 47].

A PAS sonar map is generated by placing a range dot along the transducer axis as a rotational scan is executed. With multiple readings per interrogation pulse, a PAS sonar map contains multiple dots at each interrogation angle. Rotational scans then form arcs, with isolated arcs indicating weak echoes, arc pairs moderate echoes, and arc triplets large echoes. To illustrate, Fig. 30.15 shows arcs formed by a large plane (2.3 m width by 0.6 m height) and five cylinders with different diameters, all placed at 1 m range. Examining objects at the same range eliminates effects caused by the range-dependent gain of the module.

A conventional TOF sonar map by comparison would display only the nearest arc in the PAS map for each object. Qualitatively, the arc length increases and the number of arcs increase with the echo amplitude, which is bearing dependent. The strongest reflectors produce concave arcs [30.4, 48]. This occurs because, with echo amplitudes much greater than the threshold, the threshold is exceeded near the beginning of the echo, yielding a nearly constant range reading over a significant extent over bearing. In contrast, the weakest reflectors produce convex arcs, caused by echoes whose amplitudes are comparable to the threshold. As the echo amplitude decreases the threshold is exceeded at later points along the processed waveform, producing greater range readings. This effect also appears at the edges of the arcs produced by strong reflectors.



Fig. 30.15 PAS sonar maps of six objects located at 1 m range. The sonar is located below the objects in the figure

Computing the beam pattern of the vibrating piston model [30.1], which is a reasonable approximation to the Polaroid transducer, yields the curve shown in Fig. 30.16.

This figure describes the detected echo magnitude normalized to have a maximum of 0 dB, which occurs along the beam axis. The larger echoes are much greater than the threshold, such as those produced by the plane, whose maximum amplitudes can be 44 dB relative to the threshold. The −44 dB threshold agrees with the PAS map for a plane: strong echoes (three stripes) occur within 10° of normal incidence, range readings increase due to echo amplitude reduction at ±15.6°, approximating the predicted nulls at ±14.7°, and smaller-amplitude echoes from the sidelobes are present. Weaker reflectors correspond to larger thresholds when their on-axis echoes normalize to 0 dB. The beam pattern model explains how arc length varies with object reflecting strength. The indicated thresholds were found by matching the angular beam width to the arc extent.

It is apparent that PAS maps provide information useful for solving the inverse problem, that of determining the identity of the object from the echoes. Figure 30.15 shows that PAS maps contain information about the echo amplitude. While it is true that the conventional TOF sonar maps, represented by the closest arc, can determine the object location from the arc center and can infer the echo amplitude from the arc extent, it is also true that this information is presented in a more robust way in the PAS maps. For this simple case of isolated objects, the posts can be clearly differentiated from the pole, while the corresponding conventional TOF arcs are comparable. A tenfold in-

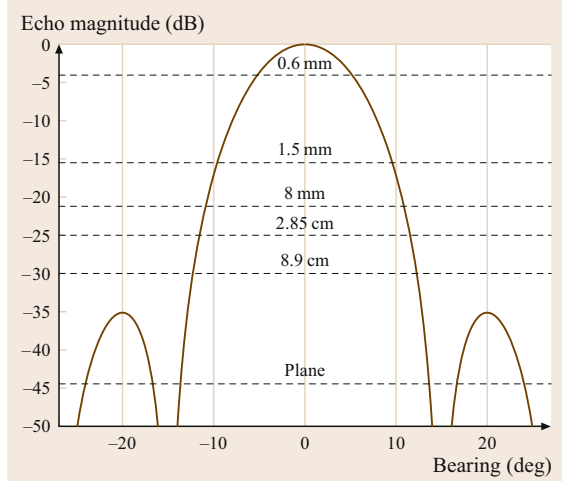


Fig. 30.16 Transmitter–receiver beam pattern. *Dashed lines* indicate the equivalent threshold level for each object

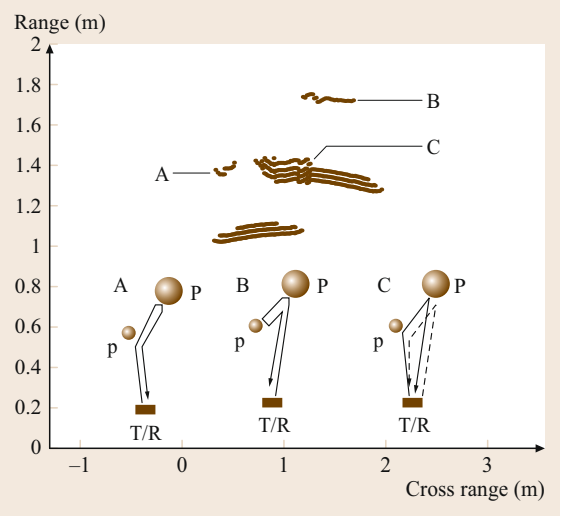


Fig. 30.17 PAS sonar map of 2.85 cm-diameter post (p) and 8.9 cm-diameter post (P). The transducer is located at (0,0). A: Artifact caused by echo originating at transmitter T, reflected by p, bouncing off P back to p, and directed toward receiver T → p → P → p → R. B: T → P → p → P → R. C: T → p → P → R and T → P → p → R

crease in post diameter yields only a modest increase in the conventional TOF arc length, while increasing the number of arcs from two to three in the PAS maps. Spike data can be processed to produce sonar images of the environment that are similar to diagnostic ultrasound B-scans [30.49]. VIDEO 315 shows a sonar B-scan of a tree.

When examining the entire echo waveform, one must account for artifacts that are produced when objects interact acoustically. Some artifacts occur after the first detected echo, so these are not a problem in TOF sonar maps [30.4], but must be addressed in interpreting PAS maps. Consider a simple environment consisting of two posts: a 2.85 cm-diameter post (p), located at $r = 1$ m and bearing 12°, and an 8.9 cm-diameter post (P), at $r = 1.3$ m and bearing −10°. The corresponding PAS map shown in Fig. 30.17 displays the echoes from the two objects plus additional echoes that illustrate two types of multiple reflection artifacts.

The first type, indicated by A and B, results when only one object is within the transducer beam. An interrogation pulse that is redirected by a reflected object must be directed back to the receiver within its beam pattern in order to be detected. The paths that do this are shown in the figure. The single-arc convex shape of A indicates that the echo has a small amplitude. This is reasonable since both reflectors are nonplanar, and hence weak.

The second type of artifact (C) shown in Fig. 30.17 occurs when both objects are within the beam pattern. This allows two distinct paths for the echoes to return to the receiver, occurring in opposite directions and doubling the artifact amplitude. With both objects lying near the beam edges, the echo amplitude is small. Since the distance traveled by these echoes is slightly greater than the range to the farther object, this artifact shows a range slightly beyond the more distant object. The superposition of these two components makes the echo from the farther object appear spread out in time. This pulse stretching explains why four arcs are observed, and at one angle five arcs. If the bearing angle between p and P was increased to exceed the beam width, this artifact would disappear.

30.10 Echo Waveform Processing

In this section pulse-echo sonar that processes sampled digitized receiver waveforms is described. These systems offer superior performance over the simple Polaroid ranging module systems described above, which report the TOF based on a threshold. Echo waveform processing does however incur the overhead of more-complex electronics and signal processing and is not readily available commercially.

30.10.1 Ranging and Wide-Bandwidth Pulses

It is shown in [30.11, 50] that the maximum-likelihood estimator (MLE) for the TOF is obtained by maximizing the correlation $\mathrm{cor}(\tau)$ between the received pulse $p(t)$ (containing Gaussian white noise) and the known pulse shape shifted by τ, $\mathrm{rec}(t-\tau)$

$$\mathrm{cor}(\tau) = \frac{\int_a^b p(t)\mathrm{rec}(t-\tau)\,\mathrm{d}t}{\sqrt{\int_a^b p^2(t)\,\mathrm{d}t \int_a^b \mathrm{rec}^2(t)\,\mathrm{d}t}}\,, \qquad (30.15)$$

where the pulse extends from time a to b. The known pulse shape at the receiver depends on the angle of transmission and reception with respect to the normals of the respective transducers. The pulse shape can be obtained by collecting a good signal to noise pulse at 1 m range at normal incidence to the receiver and transmitter and using elliptical impulse response models to obtain template pulses at angles different to normal incidence. Pulse shape also changes with range due to the dispersive properties of absorption due losses in air

transmission. These can be modeled using an estimate of the impulse response due to one meter path through air as is done in [30.11].

The correlation, $\mathrm{cor}(\tau)$ is normalized in (30.15) to be between -1 and $+1$. The correlation at the maximum thus gives a good indication of the match between the expected and actual pulse shapes and can be used to assess the quality of the TOF estimate. In practice (30.15) is used in discrete time form, where the integrals are replaced by sums of products and digital signal processors are an ideal implementation since they are highly optimized to perform this calculation [30.51, 52]. Finding the maximum correlation of (30.15) is known as template matching. To achieve an arrival time estimator with resolution smaller than the discrete time sample rate, parabolic interpolation can be used on the maximum three correlations [30.11]. Of interest is the jitter standard deviation σ_R in the TOF estimator due to receiver noise. From [30.11, 50]

$$\sigma_\mathrm{R} = \frac{\sigma_n}{B\sqrt{\sum_k \mathrm{rec}(kT_\mathrm{s})^2}}\,, \tag{30.16}$$

where the summation index k is over the entire receiver pulse sampled every T_s seconds (1 —s in [30.11, 51]), B is the bandwidth of the receiver pulse, and σ_n is the standard deviation of the receiver noise. Equation (30.16) shows that broadband high-energy pulses achieve low errors in the TOF estimator. In [30.11] this is achieved by using a 300 V pulse to excite the transmitter and achieve close to the impulse response from the device with a pulse shape similar to that shown in Fig. 30.6d.

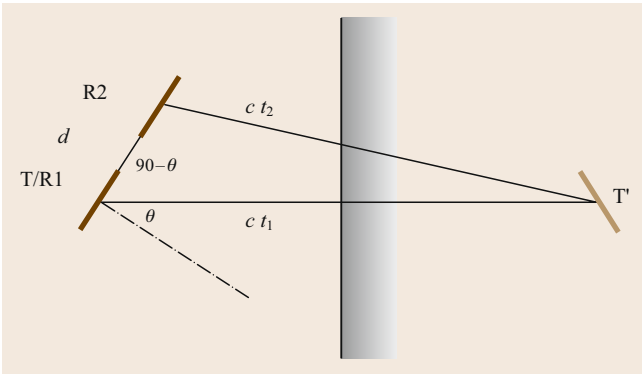


Fig. 30.18 Bearing (θ) calculation for a plane using a transceiver T/R1 and a receiver R2. T′ is the virtual image of T

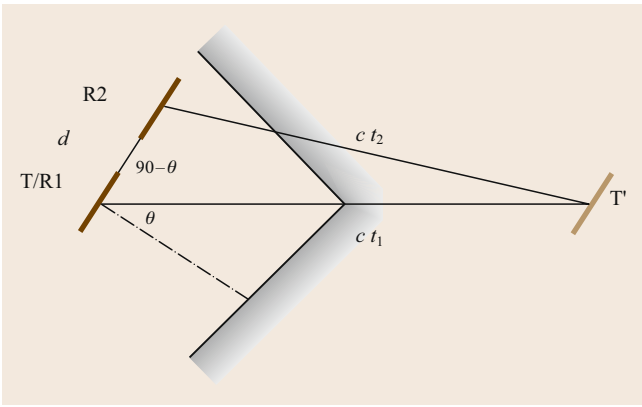


Fig. 30.19 Bearing (θ) calculation for a corner using a transceiver T/R1 and a receiver R2. T′ is the virtual image of T

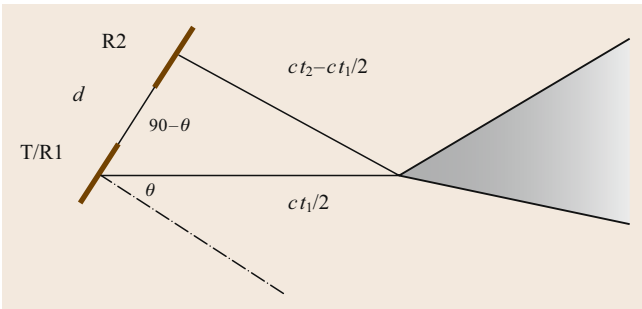


Fig. 30.20 Bearing (θ) calculation for an edge using a transceiver T/R1 and a receiver R2. No virtual image is present since the edge radiates from a point source at the edge

30.10.2 Bearing Estimation

There are many proposed methods for bearing estimation. A single transducer [30.53] can be used by exploiting the dependency of the received pulse shape on the angle of reception. This approach works for angles within one half of the beam width since the pulse shape is symmetric with respect to the transducer normal angle. Differences in zero-crossing times either side of the maximum amplitude of the pulse are used to obtain an accuracy of the order of 1°. Other single-receiver techniques rely on repeated measurements from a scan across the scene [30.54, 55] and achieve a similar level of accuracy but at much slower sensing speed since multiple readings are necessary.

Other single measurement approaches rely on two or more receivers [30.11, 12, 25]. This gives rise to a correspondence problem where data must be associated between the receivers. The closer the spacing between receivers, the simpler and more reliable the correspondence procedure. The misconception that bearing accuracy improves with larger receiver spacing ignores the correlation between measurement errors that can arise due to the measurements sharing an overlapping space of air in the propagation of the ultrasound. Due to the high accuracy of TOF estimation in [30.11] the receivers could be spaced as close as physically feasible (35 mm) and still bearing accuracies lower than any

Avidbots Ex1016
Page 802 of 2154

other systems are reported. Standard deviations of bearing errors are reported to be below 0.2° for a plane at a range of 4 m within a −10° to +10° beam width.

There are two common approaches to bearing estimation – interaural amplitude difference (IAD) [30.56] and interaural time difference (ITD) [30.11, 25, 51–53, 56]. IAD uses two receivers pointing away from each other so that an echo has a different amplitude response in each receiver's beam width. In ITD both receivers usually point in the same direction and the TOF is measured on each receiver and triangulation is applied to determine the angle of arrival. The bearing calculation is dependent on the target type, such as a plane, corner or edge; these geometries are analyzed in [30.11]. A simple arrangement with a transceiver and receiver is shown in Fig. 30.18, where T/R1 is the transceiver and R2 is the second receiver, spaced a distance d from each other.

The virtual image of the transmitter is shown as T′. The two TOFs measured on the two receivers are t_1 and t_2, and these are used to estimate the bearing angle, θ, to the plane which is the angle to the plane normal. Applying the cosine rule to the triangle R2 R1 T′ in Fig. 30.18 gives

$$\cos(90-\theta) = \sin\theta = \frac{d^2 + c^2t_1^2 - c^2t_2^2}{2dct_1} . \qquad (30.17)$$

When $d \ll ct_1$, (30.17) can be approximated by

$$\sin\theta \approx \frac{c(t_1 - t_2)}{d} . \qquad (30.18)$$

Note that any common (i. e., correlated) noise in t_1 and t_2 is removed by the difference in (30.18) and hence the correlation in noise components of the TOF cannot be overlooked in bearing estimation as described above.

The situation for a corner is shown in Fig. 30.19 and the same result applies as in (30.17).

For an edge the situation is shown in Fig. 30.20, where R1 has a TOF from T to the edge and back to R1, whilst R2 has a TOF from T to the edge and back to R2.

From the geometry, we use the same approach as in (30.17) to give

$$\sin\theta = \frac{d^2 + \frac{c^2t_1^2}{4} - c^2\left(t_2 - \frac{t_1}{2}\right)^2}{2dc\frac{t_1}{2}} = \frac{d^2 + c^2t_2(t_1 - t_2)}{dct_1} . \qquad (30.19)$$

Note that (30.19) can be approximated by (30.18) when $d \ll ct_1$.

30.11 CTFM Sonar

The continuous-transmission frequency-modulated (CTFM) sonar differs from the more common pulse-echo sonar discussed in previous sections in the transmission coding and the processing required to extract information from the receiver signal.

30.11.1 CTFM Transmission Coding

The CTFM transmitter continuously emits a varying-frequency signal, usually based on a sawtooth pattern as shown in Fig. 30.21, where the frequency is often swept through an octave every sweep cycle T.

The transmitted signal with a linearly changing frequency can be expressed as

$$S(t) = \cos\left[2\pi\left(f_H t - bt^2\right)\right] \qquad (30.20)$$

for $0 \le t < T$. The sweep cycle is repeated every T seconds as shown in Fig. 30.21. Frequency is $1/2\pi$ times the time derivative of the phase in (30.20). Note that the highest frequency is f_H and the lowest transmitted frequency is $f_H - 2bT$, where b is a constant that determines the sweep rate. We can then define the swept frequency ΔF as

$$\Delta F = 2bT . \qquad (30.21)$$

30.11.2 CTFM TOF Estimation

Echoes are generated when the transmitted wavefront encounters reflectors and are an attenuated, delayed version of the transmitted signal

$$E(t) = AS\left(t - \frac{2R}{c}\right) , \qquad (30.22)$$

where R is the range to the reflector, c is the speed of sound and A is the amplitude that may in the case of curved objects depend on the frequency of the sound at reflection.

The TOF is estimated by the two step process of demodulation and spectral analysis. Demodulation is achieved by multiplying the received signal by a copy of the transmitted signal and low pass filtering. This can best be understood in the simple case of one echo. The

signal $D(t)$ is obtained using (30.20) and (30.22)

$$\begin{aligned} D(t) &= E(t)S(t) \\ &= \frac{A}{2}\big[\cos\left(2\pi f_e t - \phi\right) \\ &\quad + \cos\left(2\pi f_u t - 2bt^2 - \phi\right)\big] \end{aligned}$$

$$\text{for } f_e = \frac{4Rb}{c}\,,$$

$$f_u = \left(2f_H + \frac{4Rb}{c}\right),$$

$$\phi = f_H\frac{2R}{c} + \frac{4bR^2}{c^2}\,, \qquad (30.23)$$

where the following trigonometric identity has been used in (30.23)

$$\cos(x)\cos(y) = \frac{1}{2}[\cos(x-y) + \cos(x+y)]\,. \qquad (30.24)$$

A low-pass filter removes frequency components above f_H and this results in the base-band signal

$$D_b(t) = \frac{A}{2}\left[\cos\left(2\pi\frac{4Rb}{c}t - \phi\right)\right]\,, \qquad (30.25)$$

which has a frequency proportional to the range R. The ranges of echoes can be extracted by examining the spectrum of D_b using, for example, a discrete Fourier transform (DFT) or the fast Fourier transform (FFT). From (30.25) for a frequency peak of f_r Hz the corresponding range R is given by

$$R = f_r\frac{c}{4b}\,. \qquad (30.26)$$

Note that the above analysis relies on excluding the receiver waveform at the start of each sweep for a *blind time* (Fig. 30.21) of $R_m/2c$, where R_m is the maximum target range. During this blind time the receiver signal is dependent on the previous sweep rather than the current sweep as assumed in the analysis above. The sweep time T needs to be much larger than this blind time for the sonar to operate effectively. The blind time can be eliminated at the expense of introducing complexity in the demodulation process as described in [30.57], where an interlaced double demodulation scheme is described.

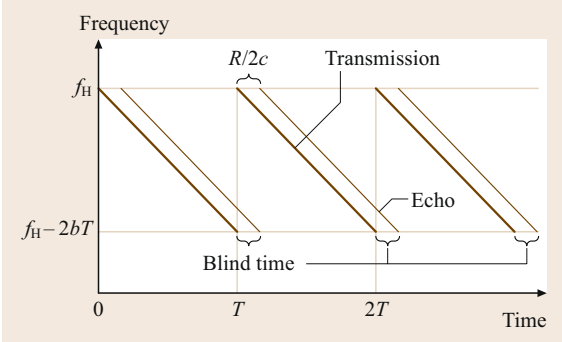


Fig. 30.21 18 Plot of frequency versus time for CTFM. The blind time applies if the shown echo corresponds to a maximum range target at R_m

30.11.3 CTFM Range Discrimination and Resolution

We define range discrimination as the separation in range of two targets that can be simultaneously detected as distinct. The range resolution is defined as the smallest increment in range that can be measured by the sonar.

Suppose that, in order to extract ranges of targets, $D_b(t)$ from (30.25) is sampled at ΔT intervals and k samples are collected before a DFT (or FFT) is performed. The frequency samples of the DFT will be $\Delta f = 1/(k\Delta T)$ apart. From (30.26), this represents a range resolution ΔR of

$$\Delta R = \frac{c\Delta f}{4b} = \frac{c}{4bk\Delta T}\,. \qquad (30.27)$$

We can relate this to the swept frequency ΔF from (30.21) as

$$\Delta R = \frac{c}{2\Delta F}\times\frac{T}{k\Delta T}\,, \qquad (30.28)$$

where the second term is the ratio of the sweep time to the spectral sample time. In order to discriminate two peaks in the DFT, they must be at least two samples apart and hence

$$\text{range discrimination} = \frac{c}{\Delta F}\times\frac{T}{k\Delta T}\,. \qquad (30.29)$$

Note that (30.28),(30.29)) show that, subject to signal-to-noise constraints, CTFM can lengthen the *data integration time* $k\Delta T$ in order to improve the range discrimination and resolution of the sonar. Also it is possible, subject to signal noise, to use interpolation techniques (e.g., parabolic interpolation) on the DFT peaks to resolve to a smaller than Δf frequency and hence improve range resolution (but not range discrimination).

30.11.4 Comparison of CTFM and Pulse-Echo Sonar

- The range resolution of pulse-echo sonar and CTFM sonar is theoretically the same given the same signal-to-noise ratios and bandwidths [30.57].

Avidbots Ex1016
Page 804 of 2154

The range discrimination in pulse-echo sonar is limited by the pulse length, where shorter pulse lengths require higher bandwidth. However in CTFM, range discrimination can be improved by increasing the data integration time, allowing more design flexibility.

- CTFM also allows for the energy of the transmitted signal to be spread evenly over time, resulting in lower peak acoustic power emission compared to pulse-echo systems with the same receiver signal-to-noise ratio. CTFM can provide a greater average power in a practical context and consequently a greater sensitivity to weak reflectors is possible.
- CTFM requires more-complex transmitter circuitry and the requirement for FFT processing on the receiver side.
- Separate transmitter and receiver transducers are necessary with CTFM, whilst pulse-echo systems can use a single transducer for both transmission and reception, resulting in restriction on the minimum range of pulse-echo sonar due to the blanking of the receiver during transmission. CTFM has no inherent restriction on minimum range.
- CTFM sonar can continuously derive range information from targets every $k\Delta T$ seconds at a delay of $R/c + k\Delta T$ compared to every $2R_m/c$ with a delay of $2R/c$ in pulse-echo sonar (ignoring processing delays in both), which may be important in real-time tracking applications.
- Other benefits of CTFM are that the number of range measurements per cycle is limited only by the range discrimination constraint of (30.28) and the signal-to-noise ratio.
- In terms of bearing estimation and classification of targets from a moving platform, short pulse-echo sonar systems like [30.27, 51] do not suffer from the CTFM data integration time required to estimate accurately the frequencies corresponding to ranges (and hence bearing). During the data integration time, the target can move with respect to the sensor and blur the measurements, making bearing estimation and classification less accurate. In short pulse-echo systems, the target is effectively sampled with a pulse of less than 100 µs, resulting in a consistent snapshot of the target.

30.11.5 Applications of CTFM

Kay [30.58, 59] developed a mobility aid for blind people using a CTFM sonar system based on a sweep of $f_H = 100$ down to 50 kHz with a sweep period of $T = 102.4$ ms. After demodulation, ranges are heard as audible tones with frequencies up to 5 kHz corresponding to ranges of up to 1.75 m. The system uses one transmitter and three receivers as shown in Fig. 30.22.

Users of the system can listen to the demodulated signal in stereo headphones corresponding to the left and right receivers, each mixed with the large central oval receiver. Higher frequencies correspond to more-distant ranges. To illustrate the sensitivity, a 1.5 mm-diameter wire is easily detectable at 1 m range – the echo produced is 35 dB above the noise floor in the system.

CTFM sonar has been used to recognize isolated plants [30.41, 60]. The advantage gained from CTFM is that extensive range and echo amplitude information is obtained from the whole plant given the spectrum of the demodulated received signal, and these echoes are obtained from an excitation across an octave of frequencies from 100 down to 50 kHz with a high signal-to-noise ratio that allows weak reflections from leaves to be sensed. This information is called the *acoustic density profile* and 19 different features are found to be useful in classifying the plants, such as the number of range cells above a threshold in amplitude, the sum of all range cells, the variation about the centroid, the distance from the first to the highest amplitude cell, and the range over which reflections are detected. With a population of 100 plants, an average of 90.6% correct pairwise classification was obtained using a statistical classifier.



Fig. 30.22 Aid for blind people – the small oval transducer is the transmitter and the other three components are receivers. The large oval receiver provides high resolution, enabling fixation by users' fine neck control (after *Kay* [30.58])

Avidbots Ex1016
Page 805 of 2154

Scanning CTFM with a singe transmitter and single receiver has been successfully applied to mapping of indoor environments that include smooth and rough surfaces [30.55] with bearing errors on the order of 0.5° for smooth surfaces and higher for edges. The classification uses amplitude information that is normalized with range using a fixed attenuation constant of sound. In practice this attenuation constant varies with temperature and humidity and needs to be calibrated before each experiment for consistent results. Greater robustness, speed, and accuracy has been demonstrated with TOF methods of classification that require at least two transmitter positions and two receivers as described in [30.11, 51]. CTFM could be applied to array systems to achieve higher sensitivity to weak targets than the existing pulse-echo systems.

CTFM has been employed in three binaural systems [30.12] where a rigorous theoretical and experimental comparison of these ultrasonic sensing systems based on different range and bearing estimators is made. *Stanley* [30.12] also contains detailed engineering design information of CTFM sonar systems. The conclusion is that CTFM can insonify large areas due to its higher average power transmissions and consequently good signal to noise performance. The use of autoregressive estimators for spectral lines in the demodulated signal were found to provide better resolution than the DFT. The interaural distance and power difference CTFM approaches provided state-of-the-art performance except that the pulse-echo approach in [30.11] using a high-energy short pulse was found to be a *factor of six to eight* times superior in bearing precision.

30.12 Multipulse Sonar

This section examines sonar systems that employ more than one pulse in the transmitter(s). The main motivations are interference rejection and on-the-fly classification. Multipulse sonar has also been used to generate a better signal-to-noise ratio by creating longer transmitted pulse sequence using Barker codes [30.61]. The autocorrelation of a Barker code gives a narrow peak with low autocorrelation away from the central lobe. The matched filter then gives rise to pulse compression that averages noise over a longer time period.

30.12.1 Interference Rejection

External acoustic noise, such as compressed air, is a source of sonar interference. Sonar systems attempt to reduce the effects of external interference by filtering the signal and the optimal filter is the matched filter where the impulse response is the time reversal of the pulse shape that is expected. Since a time-reversed convolution is a correlation, the matched filter then acts as a correlation with the expected pulse shape as discussed in Sect. 30.10. Approximations to matched filtering can be designed based on a bandpass filter with a frequency response that is similar to the spectrum of the expected receiver pulse. CTFM systems allow robust suppression of external interference by employing a matched filter across a broad range of frequencies contained in the continuous chirp transmission.

When more than one sonar system operates in the same environment, the transmitted signal from one sonar system can be received by another, causing crosstalk errors. This is particularly evident in classical sonar rings constructed from Polaroid ranging modules. Error-eliminating rapid ultrasonic firing strategies have been developed [30.62] and are claimed to remove most of this interference and allow faster operation of these sonar rings.

More-sophisticate coding of transmitted pulse(s) has been employed [30.23, 63–66] to allow rejection of external interference and crosstalk. One difficulty with multiple transmitted pulses over a greater time period than a single pulse is that target clutter can produce many overlapping pulses at the receivers that are difficult to unravel and interpret, and the sonar range discrimination can be compromised.

30.12.2 On-the-Fly Target Classification

Target classification into planes, cylinders, and edges has been achieved by deploying a single transmitter and three receivers [30.25] using a single measurement cycle. At least two transmitters are required to differentiate planes from concave right-angled corners [30.11] where a two transmitter arrangement is used to classify targets into planes, corners, and edges in two successive measurement cycles. The method of classification can be understood with virtual images and mirrors, since specular sonar reflections occur. Looking into a plane mirror gives an image that is left–right reversed compared to looking into a right-angled mirror. An edge is analogous to observing a high-curvature specular surface, such as a polished chair leg, where the whole image is compressed to a point. Sonar classification exploits the difference in bearing angles to a target from

two transmitters to classify as follows: a positive difference δ indicates a plane, a negative δ value indicates a corner, and zero difference an edge, where the angle δ depends on the sensor geometry and target range. More sophistication can be added by using range measurements in addition to bearing, with maximum-likelihood estimation.

This arrangement [30.11] was refined to work with just one measurement cycle of around 35 ms to 5 m range: hence the term *on-the-fly* in [30.51]. This on-the-fly approach uses pulses fired at a precise time difference ΔT and 40 mm apart from two transmitters with two further receivers completing a square. ΔT is usually around 200 µs but can vary randomly from cycle to cycle to achieve interference rejection (both crosstalk and environmental) with identical sonar systems. Classification is performed simultaneously in one measurement cycle. The sensor achieves high accuracy in range and bearing with robust classification by exploiting the tight correlation between TOF jitter in the different transmitter to receiver paths due to the close temporal and spatial arrangement. The sensor has been deployed for large-scale mapping in [30.67].

30.13 Sonar Rings and Arrays

Since sonar only detects objects lying within its beam, a common means to scan the entire environment outside the robot is to use an array of sonars, or a *ring* [30.68].

30.13.1 Simple Ranging Module Rings

The most common is the Denning ring that contains 24 sonars equally spaced around the robot periphery. This 15° spacing allows some overlap in the sonar beams so at least one of the sonars will detect a strong reflecting object. The sonars in the ring are typically employed sequentially one at a time. Using a 50 ms probing pulse period to reduce false readings a complete environmental scan is accomplished every 1.2 s. This sample time is adequate for a translate-and-stop operation in research settings, but may be too slow for a continually moving robot. A robot moving at 1 m/s may not detect an object with sufficient warning to prevent a collision. Some researchers propose simultaneously employing sonars on opposite ends of the ring to speed up acquisition times, while others also reduce the probing pulse period and attempt to identify artifacts.

30.13.2 Advanced Rings

Yata et al. [30.53] developed a 32 cm-diameter sonar ring with 30 transmitters and 30 receivers placed alternately. Murata piezoelectric MA40S4R wide-angle transducers are used to enable overlapping reception of echoes produced by firing all the transmitters simultaneously. An axial symmetrical exponential horn structure is used to narrow the beam shape of the transmitters vertically to avoid reflections from the floor. Received signals are compared with a decaying threshold to produce a 1 bit digitized sampled signal without rectification. Bearing is estimated from the leading edge of echoes and an error standard deviation of 0.4° is reported for ranges up to 1.5 m.

A sonar ring with seven digital signal processors (DSPs) [30.52, 69, 70] that uses 24 pairs of 7000 series Polaroid transducers consisting of a transceiver and receiver has been developed (Fig. 30.23). Each pair can derive range and accurate bearing information using template-matching digital signal processing (Sect. 30.10) on each of the two receiver channels, which are sampled at 250 kHz with 12 bit analog to digital converters. In total eight receiver channels are processed per DSP. All transceivers are fired simultaneous to enable full surrounding sensing of the environment approximately 11 times a second to a 6 m range with experimentally validated range and bearing accuracies to smooth targets of 0.6 mm and 0.2°, respectively. To suppress interference between neigh-

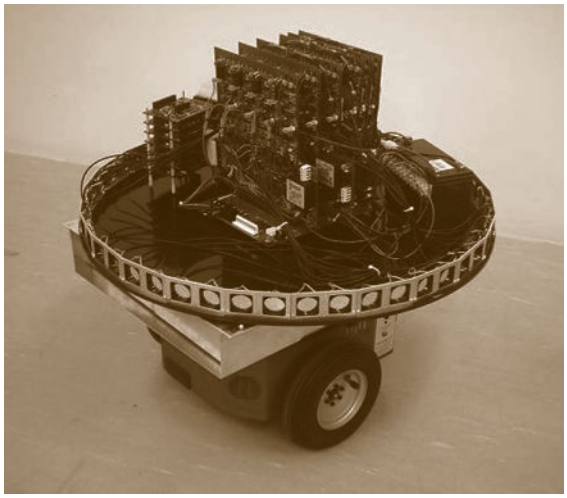


Fig. 30.23 DSP sonar ring hardware

boring pairs, two different transmitted pulse shapes are employed in an interleaved fashion around the perimeter of the ring. The pulse shapes are derived from two and three cycles of 65 kHz excitation. The DSP sonar ring allows rapid and accurate wall following, map building, and obstacle avoidance due to the high repetition and accurate range and bearing sensing. The beam width of the transducer pairs allows full 360° coverage with respect to smooth specular targets to a range of 3 m. An example of the DSP sonar ring producing a feature for a simultaneous localization and mapping (SLAM) map is shown in Fig. 30.24.

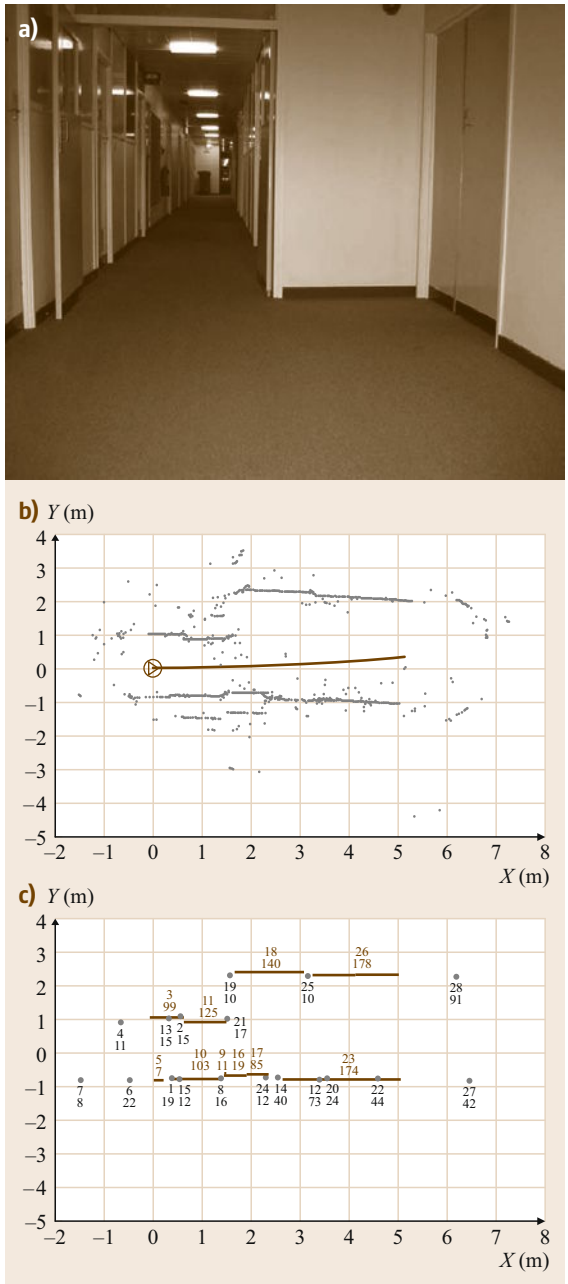


Fig.30.24a–c DSP sonar ring mapping an indoor environment (**a**), raw data (**b**) and SLAM feature map with feature number and number of associations shown as numbers (**c**). The sonar ring is moving at 10 cm/s with a 11.5 Hz sampling rate

30.13.3 Sonar Rings with FPGA Hardware Processing

One of the limitations of DSP based implementation of the signal processing arrangments such as [30.52, 69, 70] is that insufficient processing is available to perform matched filtering on the complete receiver echo signals in a multi-channel sonar ring. That is selected segments of the echo are processed where the signal exceeds a threshold in order to limit the computational load on the DSP processors. Once the full echo has arrived, matched filtering is then applied to these segments. More recent work [30.71, 72] describes a sonar ring that processes the complete echo signals from 48 receivers as the signal arrives with matched filtering. Complete processing of the entire receiver signal allows weak echoes to be detected that would be below the threshold of segmentation in [30.52, 69, 70]. This is achieved using a single field programmable gate array (FPGA) configured with a custom data path hardware design performing 4.9 Giga-arithmetic operations per

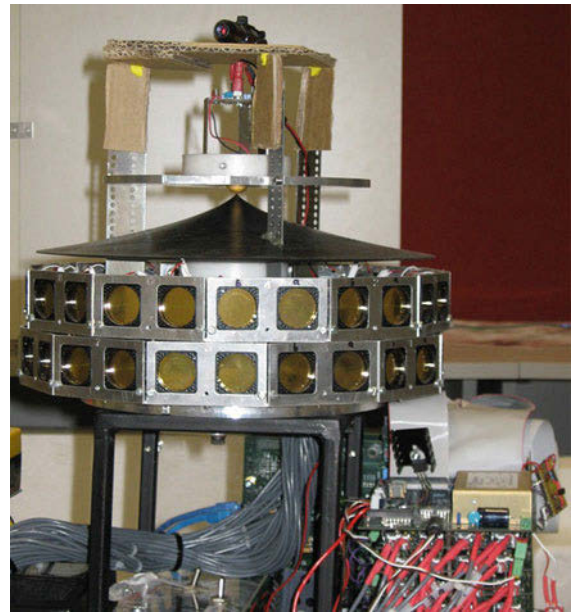


Fig. 30.25 FPGA based sonar ring

second. VIDEO 313 shows DSP sonar tracking an object. Range and bearing measurements are obtained at a rate of 30 Hz to 4 m range with full surrounding 360° angular coverage.

A photo of the FPGA sonar ring is shown in Fig. 30.25. Note that two layers of receivers are employed to reduce the radius of the 48 receiver ring. The other notable feature is the use of a single transmitter at the center of the ring. The transmitter is a hemispheric tweeter ESTD01 from Murata and a conical parabolic reflector to distribute the transmitter pulse uniformly in all directions horizontally [30.73]. Two cycles of a 48 kHz sine wave form the transmitter pulse. The transmitter can be modelled as point source and the pulse shape is approximately invariant with respect to received angle for each pair of receivers. This simplifies the matched filtering since the template for each receiver pair does not depend on the receiver angle. However there is some variation of pulse shape between different pairs in practice and as a function of range. The template match correlation processing hardware has been designed to allow dynamic switching of the template at predetermined ranges to account for the range variation of the pulse shape.

30.13.4 Double Refresh Rate Sonar Ring

Sonar rings have been limited in their firing rate by waiting for maximum range echoes to return before a consecutive firing of the transmitter. For example, the FPGA sonar ring presented in [30.71] is limited to 30 Hz refresh rate by this constraint. A double refresh rate FPGA sonar ring is presented in [30.74] that operates at 60 Hz for a 5.7 m range. To achieve a double refresh rate, the next sonar transmission is scheduled by selecting a transmit time from a random set so as to minimize interference with predicted echo arrival times. Two matched filters are applied to the receiver signal based on the two possible transmit times. Echoes are associated with a transmit epoch with the following criteria:

- The arrival time must be consistent with previous echoes since this is bounded by the sensor to obstacle relative speed
- The pulse energies should be within 50% of a previous echo from the same obstacle, and finally
- The template match correlation must be acceptable for the assumed range.

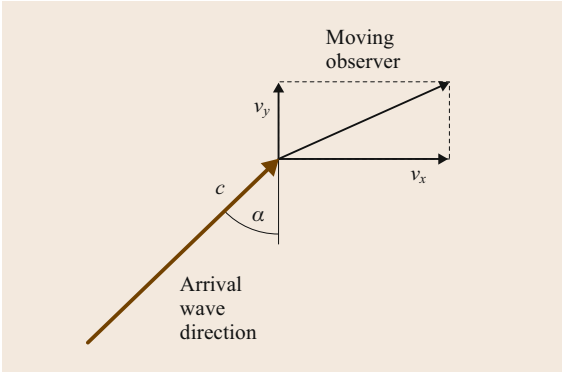
30.13.5 Sparse 3-D Arrays

The work by *Steckel* et al. [30.75] presents a sonar system that borrows beamforming theory from the design of antenna arrays. The sonar array is based on a sparse random array of 32 condensor omnidirectional ultrasonic microphones and a single Polaroid 7000 acting as a transmitter. The microphone signals are sampled at a maximum rate of 500 kHz and processed by an FPGA to produce a measurement repetition rate of 12 Hz. The transmitter waveform is a carefully controlled hyperbolic chirp of 100 down to 20 kHz in 3 ms. Matched filtering and beamforming processing are applied to the received signals to generate an energy scape – the distribution of energy from reflectors from discrete directions of around 1° resolution. Experiments show the system can discriminate 20 mm diameter poles at 800 mm with 5° separation. The minimum angular separation is related to the width of the mainlobe of the spatial array filters. Azimuth and elevation standard deviations of errors are reported as 1.1 and 0.6° respectively for a 80 mm sphere at 1.5 m. The advantage of the system is its ability to discriminate overlapping echoes in cluttered environments.

30.14 Motion Effects

When a sensor moves with respect to its targets, sonar measurements are effected. For example, a sonar sensor moving at a speed of 1% of the speed of sound (around 3.4 m/s) will experience errors of the order of 0.6° for some bearing measurements. The effects of linear velocity on the TOF and reception angle are dependent on the target type and hence for motion compensation to be meaningful a target classification sensor is needed. We consider the classical plane, edge, and corner target types in this section. Rotational motion effects are discussed in [30.27] where it is shown that very high speeds of rotation are necessary to give rise to a small bearing error (e.g., 0.1° error for approximately 1700 deg/s). Narrowing of the effective beam width is another effect of high rotation speeds of a sonar sensor.

The sensor is assumed to transmit from a point labeled T and receiver measurements are referenced to this position on the sensor. However, due to the motion of the sensor, the ground referenced position R at the time of reception of the echo moves from T over the course of the TOF. For a linear velocity, the distance between T and R is TOF$\times v$, where v is the magnitude of the sensor velocity vector relative to the ground, with components v_x and v_y parallel to their respective



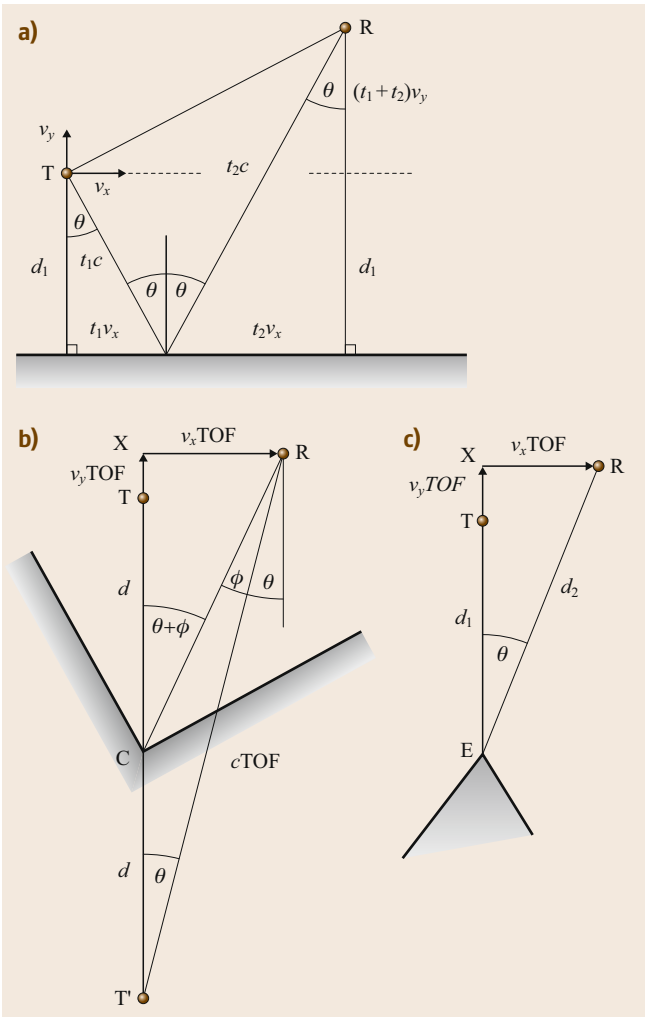


Fig.30.27a–c Observing a target from a moving sensor. T is the position of the transmitter and R is where the echo is received at the end of the TOF. The target is a plane in **(a)**, a corner in **(b)**, and an edge in **(c)**

Fig. 30.26 Observation of arriving wave from a moving observer ◀

coordinate axes. The expressions derived for linear motion apply to any sonar sensor, since only the physics of sound propagation and reflection are used. All targets are assumed to be stationary.

This section is based on [30.27], where further experimental work not included here can be found.

30.14.1 Moving Observation of a Plane

A plane target reflects the transmission from position T to R as shown in Fig. 30.26a. The TOF is broken up into two parts: t_1 is the time of propagation to the plane and t_2 from the plane to the receiver R. Here we derive the effect of linear motion on the TOF $= t_1 + t_2$ and the angle of reception θ all taken from the view of a *stationary observer*. A moving observer is discussed below.

From the right-angle triangle on the left of Fig. 30.27 a, we have

$$\sin\theta = \frac{v_x}{c} \text{ and } \cos\theta = \sqrt{1-\left(\frac{v_x}{c}\right)^2} \qquad (30.30)$$

and also

$$\cos\theta = \frac{d_1}{t_1 c} \implies t_1 = \frac{d_1}{c\cos\theta}\,. \qquad (30.31)$$

From the right-angled triangle on the right of Fig. 30.27 a, we have

$$\cos\theta = \frac{(t_1+t_2)\,v_y + d_1}{t_2 c}$$

$$\implies t_2 = \frac{(t_1+t_2)\,v_y + d_1}{c\cos\theta}\,. \qquad (30.32)$$

The TOF is obtained by adding (30.31) and (30.32) and then substituting (30.31) giving

$$\text{TOF} = \left(\frac{2d_1}{c}\right)\frac{1}{\sqrt{1-\frac{v_x{}^2}{c}-\frac{v_y}{c}}}\,. \qquad (30.33)$$

The first factor in (30.33) represents the stationary TOF. The second factor approaches unity as the velocity approaches zero.

30.14.2 Moving Observation of a Corner

Figure 30.27 b shows the situation for a corner with the virtual image of T is shown as T′. From the right-angled triangle T′XR

$$c^2\text{TOF}^2 = (2d_1 + v_y\text{TOF})^2 + v_x^2\text{TOF}^2\,, \qquad (30.34)$$

Avidbots Ex1016
Page 810 of 2154

which gives

$$\mathrm{TOF} = \frac{2d_1}{c}\left(\frac{\sqrt{1-\left(\frac{v_x}{c}\right)^2}+\frac{v_y}{c}}{1-\left(\frac{v}{c}\right)^2}\right), \tag{30.35}$$

where $v^2 = v_x^2 + v_y^2$. The left-hand term of (30.35) is the stationary TOF and the right-hand term approaches unity for small velocities. The angle ϕ in Fig. 30.27 b is the angle deviation due to motion as reference by a stationary observer. From the triangles T′XR and CXR

$$\tan\theta = \frac{v_x\mathrm{TOF}}{2d_1 + v_y\mathrm{TOF}} \quad \text{and}$$
$$\tan(\theta+\phi) = \frac{v_x\mathrm{TOF}}{d_1 + v_y\mathrm{TOF}}\,. \tag{30.36}$$

From (30.36), we have

$$\tan(\theta+\phi) = \left(2 - \frac{v_y\mathrm{TOF}}{d_1 + v_y\mathrm{TOF}}\right)\tan\theta \tag{30.37}$$

and solving for $\tan\phi$ yields

$$\tan\phi = \tan\theta\left(\frac{1-\sin^2\theta}{\frac{v_y\mathrm{TOF}}{d_1}+1+\sin^2\theta}\right) = \left(\frac{v_x}{\frac{2d_1}{\mathrm{TOF}}+v_y}\right)\left(\frac{1-\sin^2\theta}{\frac{v_y\mathrm{TOF}}{d_1}+1+\sin^2\theta}\right). \tag{30.38}$$

For $v_x, v_y \ll c$, $\sin\theta \ll 1$ and $2d_1/\mathrm{TOF} \approx c$ we can approximate (30.38) as

$$\phi \approx \frac{v_x}{c}\,. \tag{30.39}$$

30.14.3 Moving Observation of a Edge

Since an edge reradiates the incoming ultrasound from an effective point source, the reception angle with respect to a stationary observer is unaffected by motion as shown in Fig. 30.27 c. The TOF is affected due to the motion moving the receiving position. From the right-angled triangle XER, $d_2^2 = (d_1 + v_y)^2 + v_x^2\mathrm{TOF}^2$ and $d_1 + d_2 = c\mathrm{TOF}$ leads to

$$\mathrm{TOF} = \frac{2d_1}{c}\left(\frac{1+\frac{v_y}{c}}{1-\frac{v^2}{c^2}}\right) \approx \frac{2d_1}{c}\left(1+\frac{v_y}{c}\right), \tag{30.40}$$

where the approximate holds in (30.40) for $v \ll c$.

30.14.4 The Effect of a Moving Observation on the Angle of Reception

The expressions for the reception angle in the previous sections are based on an observer that is stationary with respect to the propagating medium air. In practice the observer is the sensor, and is moving with a velocity v. Suppose that the sonar wave arrives at an angle α relative to the air, as shown in Fig. 30.26.

The velocity components of the wavefront relative to the observer, w_x and w_y, are as follows

$$w_x = c\sin\alpha - v_x \text{ and } w_y = c\cos\alpha - v_y\,. \tag{30.41}$$

From (30.41) the observed angle of arrival, β is

$$\tan\beta = \frac{c\sin\alpha - v_x}{c\cos\alpha - v_y} = \frac{\sin\alpha - \frac{v_x}{c}}{\cos\alpha - \frac{v_y}{c}}\,. \tag{30.42}$$

30.14.5 Plane, Corner, and Edge Moving Observation Arrival Angles

In this section the arrival angles (in radians) for each target type are summarized and approximated for speeds expected of a mobile robot. The speed is assumed to be less than a few percent of the speed of sound (typically 340 m/s at room temperature). These effects have been observed experimentally at speeds of up to 1 m/s [30.27].

Equations (30.41) and (30.30) cancel exactly, and for a plane the arrival angle relative to the sensor is exactly zero

$$\beta_{\mathrm{plane}} = 0\,. \tag{30.43}$$

This can be explained by noting that the forward wave velocity component is always the same as the sensor's as reflection preserves this component.

For a corner the angle ϕ results in a wavefront that appears to be displaced in the same direction as the sensor motion from the real corner direction, as can be seen in Fig. 30.27 b. The effect of the moving observer doubles this effect as seen by (30.41) and (30.39)

$$\beta_{\mathrm{corner}} \approx -\frac{2v_x}{c}\,. \tag{30.44}$$

For an edge the result is due to the observer only

$$\beta_{\mathrm{edge}} \approx \tan^{-1}\left(\frac{0-\frac{v_x}{c}}{\cos\alpha - \frac{v_y}{c}}\right) \approx -\frac{v_x}{c}\,. \tag{30.45}$$

Avidbots Ex1016
Page 811 of 2154

30.15 Biomimetic Sonars

The success of biosonars, bats and dolphins [30.76], has led researchers to implement sonars based on biosonar morphology, strategy, nonlinear processing and cleverness [30.77]. The capabilities exhibited by biosonars have caused researchers to examine biomimicking (*biomimetic*) systems.

Biosonar morphology typically has a single transmitter and a pair of receivers. Bats transmit sound pulses through the mouth or nose, while dolphins transmit through a melon. The two receivers correspond to ears that permit binaural processing. Mimicking binaural hearing has led to small arrays that localize objects [30.8] and scanning strategies [30.78]. VIDEO 311, VIDEO 316, VIDEO 317 show biomemetic sonars in action. Movable pinnae observed in bats have motivated research in receivers that rotate [30.79, 80] and deform [30.81]. VIDEO 312 shows pinna deformation. Figure 30.28 shows one such example.

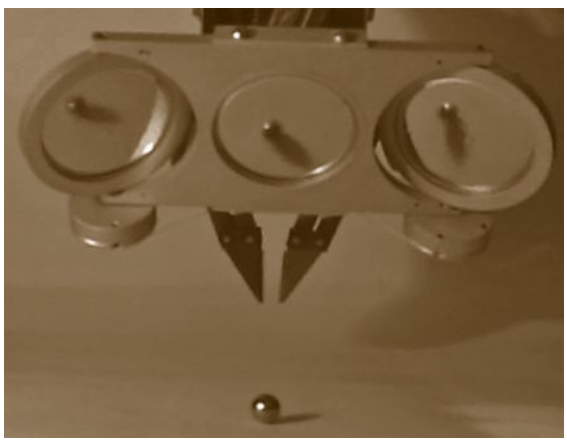


Fig. 30.28 Biomimetic configuration sonar with center transmitter flanked by receivers that rotate

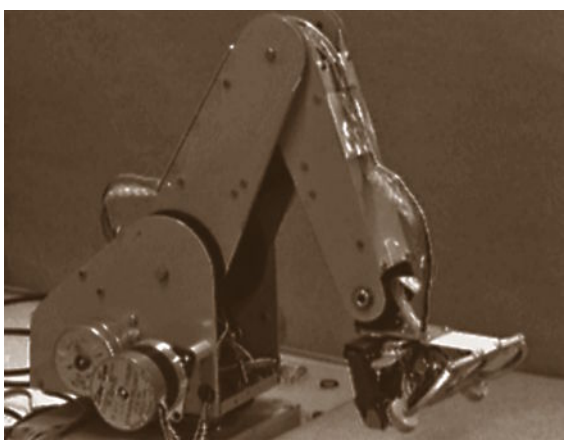


Fig. 30.29 Biomimetic sonar mounted on the end of a robot arm

Rotating the receivers so their axes fall onto the reflecting object not only increases the detected echo amplitude, but also its bandwidth, both effects improving the ability to classify an object.

Biosonar strategy provides clues for successful object localization. It is well known that the object location within the transducer beam affects the echo waveform and complicates the inverse problem of object classification [30.10, 82]. Dolphin movies show that they maneuver to position an object at a repeatable location and range, guided by binaural echo processing. This has motivated a dolphin-mimicking movable sonar positioned at the end of a robot arm for object classification [30.10, 82], as shown in Fig. 30.29.

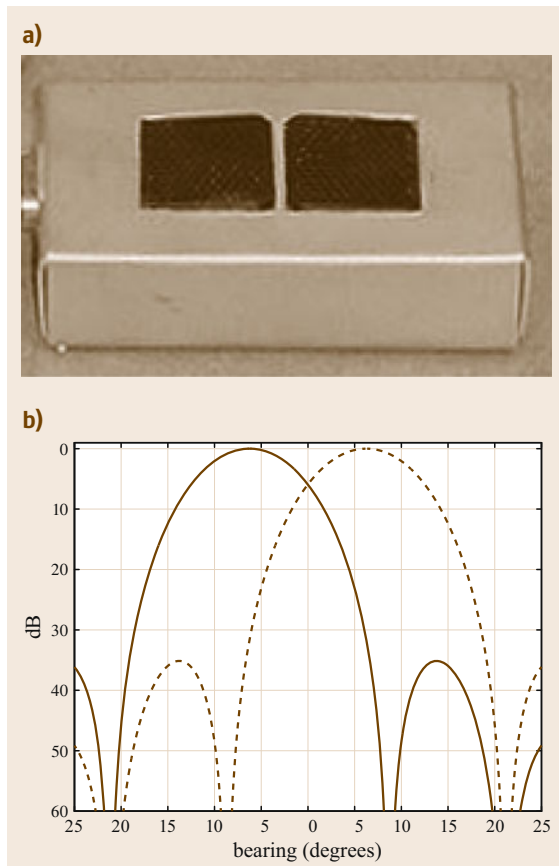


Fig.30.30a,b Binaural vergence sonar **(a)** and beam patterns **(b)** along the sonar axis with vergence angle $\alpha = 8^0$: *solid-left* transducer, *dashed-right* transducer

This system was able to differentiate reliably the head and tail side of a coin, but only after introducing a scan in elevation to accommodate for the lack of such positioning afforded by binaural hearing. The idea for a scan over elevation was motivated by the nodding motion that dolphins exhibit when searching for prey lying under the sand.

Another useful strategy, suggested by probing pulses emitted by bats, is processing echo sequences. As an extension to the conventional stop-and-scan operation of most sonars, sonar data were acquired while the sonar was moving along piecewise linear paths to reveal hyperbolic trends, similar to acoustic flow [30.83]. Matching data to hyperbolic trends permits the estimation of the passing range, which is useful for collision avoidance and passing through narrow openings [30.83].

Most sonar systems use classical estimation procedures involving correlation detection and spectrum analysis. The cochlear model has led to multiple band-pass filters to process wide-band pulses for environmental landmark classification [30.43]. The action potential spikes observed in the biological nervous system also suggest neuromorphic processing based on coincidence detection. The sparse information provided by conventional TOF measurements motivated sonar detectors that provided complete echo waveform information from multiple detections that result in spike-like data [30.24, 84]. Applying temporal and spacial coincidence to spike data has led to reverberation artifact recognition [30.36] and passing-range estimation [30.85]. VIDEO 302 shows a simulation of side-looking sonar passing two targets. VIDEO 314 shows camera view of mobile robot traveling down corridor. VIDEO 303 shows sonar spike map of corridor.

One shortcoming of sonar is its relatively wide beam that limits the accuracy of the reflector bearing estimate. Biosonars have two ears that assist in echolocation, with a strategy that is surprising for engineers. Figure 30.30 shows a biomimetic binaural sonar described in [30.86] that employs two Polaroid 6500 ranging modules connected to series-7000 ultrasound transducers (dia $d = 2.5$ cm), designated as *right* and *left*. Both transducers transmit emissions simultaneously, but each makes a TOF reading. The transducers are separated by D cm, with D minimized so that the incident echo wavefront is as similar as possible across both transducers. If both transducers are oriented parallel to the sonar axis, the TOF difference for an echo from an object at bearing θ is

$$\Delta\text{TOF} = \frac{D}{c}\sin\theta;, \tag{30.46}$$

which is small for small D.

While most sensor engineers would maximize signal strength by having both transducers point at the object, biological sonars trade off signal strength for improved localization by directing the ears away from the object with vergence angle α. This vergence angle spreads the sensitivity patterns away from the sonar axis, as shown in Fig. 30.30. As was shown in Fig. 30.12, this enhances ΔTOF by increasing the echo amplitude difference. VIDEO 301 shows vergence sonar in operation.

Such a vergence sonar has several applications for mobile robots. By noting when both TOFs are nearly coincident, the binaural vergence sonar can detect when a reflecting object is within $\pm 0.1^\circ$ of the sonar orientation. This sonar is useful for mobile robot navigation because, in addition to a range measurement, it indicates whether an obstacle is to the left or right of the intended path. VIDEO 295 shows mobile robot equipped with multiple vergence sonars. Also, the occlusion problem associated with sonar sensing is solved by placing vergence sonars on the outside limits of a robot to determine if an opening lies to the outside of the robot dimensions, thus allowing the robot to pass through narrow openings.

Such biomimetic techniques provide insights into the information content present in echoes and the type of sensing tasks for which sonar is best suited.

30.16 Conclusions

Sonar is a useful, inexpensive, low-power, light-weight and simple ranging sensor for robotics applications that can provide accurate object localisation. For sonar to be effectively employed, understanding its physical principles and implementation is important and these topics are covered in the early sections of this chapter. Various approaches to sonar sensing are highlighted from simple single-transducer ranging to more sophisticated multi-transducer and multi-pulse configurations with associated signal processing requirements. Sophisticated sonars are capable of measuring target range and angle accurately as well as classifying targets, rejecting interference, and compensating for motion. Sonar rings provide surrounding environmental coverage, and CTFM systems improve the sensitivity for detecting small reflectors. Research is ongoing in areas such as

signal and data processing, sonar map building, sonar configurations, transducer technology and biomimetic sonar that draw inspiration from biological sonar systems, such as used by bats and dolphins.

Video-References

VIDEO 295 Sonar guided chair at Yale
available from http://handbookofrobotics.org/view-chapter/30/videodetails/295

VIDEO 301 Vergence sonar
available from http://handbookofrobotics.org/view-chapter/30/videodetails/301

VIDEO 302 Side-looking TOF sonar simulation
available from http://handbookofrobotics.org/view-chapter/30/videodetails/302

VIDEO 303 Side-looking multi-pulse sonar moving down cider-block hallway
available from http://handbookofrobotics.org/view-chapter/30/videodetails/303

VIDEO 311 Antwerp biomimetic sonar tracking complex object
available from http://handbookofrobotics.org/view-chapter/30/videodetails/311

VIDEO 312 Biological bat ear deformation in sonar detection
available from http://handbookofrobotics.org/view-chapter/30/videodetails/312

VIDEO 313 Monash DSP sonar tracking a moving plane
available from http://handbookofrobotics.org/view-chapter/30/videodetails/313

VIDEO 314 Side-looking sonar system traveling down hallway (camera view)
available from http://handbookofrobotics.org/view-chapter/30/videodetails/314

VIDEO 315 B-scan image of indoor potted tree using multi-pulse sonar
available from http://handbookofrobotics.org/view-chapter/30/videodetails/315

VIDEO 316 Antwerp biomimetic sonar tracking single ball
available from http://handbookofrobotics.org/view-chapter/30/videodetails/316

VIDEO 317 Antwerp biomimetic sonar system tracking two balls
available from http://handbookofrobotics.org/view-chapter/30/videodetails/317

References

30.1 L.E. Kinsler, A.R. Frey, A.B. Coppens, J.V. Sanders: *Fundamentals of Acoustics* (Wiley, New York 1982)

30.2 R.C. Weast, M.J. Astle (Eds.): *CRC Handbook of Chemistry and Physics*, 59th edn. (CRC, Boca Raton 1978)

30.3 J. Borenstein, H.R. Everett, L. Feng: *Navigating Mobile Robots* (Peters, Wellesley 1996)

30.4 R. Kuc, M.W. Siegel: Physically-based simulation model for acoustic sensor robot navigation, IEEE Trans. Pattern Anal. Mach. Intell. **9**(6), 766–778 (1987)

30.5 SensComp: 7000, http://www.senscomp.com (2007)

30.6 H.H. Poole: *Fundamentals of Robotics Engineering* (Van Nostrand, New York 1989)

30.7 J.E. Piercy: *American National Standard: Method for Calculation of the Absorption of Sound by the Atmosphere*, Vol. ANSI S1-26-1978 (Acoust. Soc. Am., Washington 1978)

30.8 B. Barshan, R. Kuc: A bat-like sonar system for obstacle localization, IEEE Trans. Syst. Man Cybern. **22**(4), 636–646 (1992)

30.9 R. Kuc: Three dimensional docking using qualitative sonar. In: *Intelligent Autonomous Systems IAS-3*, ed. by F.C.A. Groen, S. Hirose, C.E. Thorpe (IOS, Washington 1993) pp. 480–488

30.10 R. Kuc: Biomimetic sonar locates and recognizes objects, J. Ocean Eng. **22**(4), 616–624 (1997)

30.11 L. Kleeman, R. Kuc: Mobile robot sonar for target localization and classification, Int. J. Robotics Res. **14**(4), 295–318 (1995)

30.12 B. Stanley: A Comparison of Binaural Ultrasonic Sensing Systems, Ph.D. Thesis (University of Wollongong, Wollongong 2003)

30.13 Material Systems Inc.: http://www.matsysinc.com/

30.14 F.L. Degertekin, S. Calmes, B.T. Khuri-Yakub, X. Jin, I. Ladabaum: Fabrication and characterization of surface micromachined capacitive ultrasonic immersion transducers, J. Microelectromech. Syst. **8**(1), 100–114 (1999)

30.15 B. Barshan, R. Kuc: Differentiating sonar reflections from corners and planes by employing an intelligent sensor, IEEE Trans. Pattern Anal. Mach. Intell. **12**(6), 560–569 (1990)

30.16 A. Freedman: A mechanism of acoustic echo formation, Acustica **12**, 10–21 (1962)

30.17 A. Freedman: The high frequency echo structure of somae simple body shapes, Acustica **12**, 61–70 (1962)

30.18 Ö. Bozma, R. Kuc: A physical model-based analysis of heterogeneous environments using sonar – ENDURA method, IEEE Trans. Pattern Anal. Mach. Intell. **16**(5), 497–506 (1994)

30.19 Ö. Bozma, R. Kuc: Characterizing pulses reflected from rough surfaces using ultrasound, J. Acoust. Soc. Am. **89**(6), 2519–2531 (1991)

Avidbots Ex1016
Page 814 of 2154

30.20 P.J. McKerrow: Echolocation – from range to outline segments. In: *Intelligent Autonomous Systems IAS-3*, ed. by F.C.A. Groen, S. Hirose, C.E. Thorpe (IOS, Washington 1993) pp. 238–247

30.21 J. Thomas, C. Moss, M. Vater (Eds.): *Echolocation in Bats and Dolphins* (University of Chicago Press, Chicago 2004)

30.22 J. Borenstein, Y. Koren: Error eliminating rapid ultrasonic firing for mobile robot obstacle avoidance, IEEE Trans. Robotics Autom. **11**(1), 132–138 (1995)

30.23 L. Kleeman: Fast and accurate sonar trackers using double pulse coding, Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS) (1999) pp. 1185–1190

30.24 R. Kuc: Pseudo-amplitude sonar maps, IEEE Trans. Robotics Autom. **17**(5), 767–770 (2001)

30.25 H. Peremans, K. Audenaert, J.M. Van Campenhout: A high-resolution sensor based on tri-aural perception, IEEE Trans. Robotics Autom. **9**(1), 36–48 (1993)

30.26 A. Sabatini, O. Di Benedetto: Towards a robust methodology for mobile robot localization using sonar, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (1994) pp. 3142–3147

30.27 L. Kleeman: Advanced sonar with velocity compenstation, Int. J. Robotics Res. **23**(2), 111–126 (2004)

30.28 A. Elfes: Sonar-based real world mapping and navigation, IEEE Trans. Robotics Autom. **3**, 249–265 (1987)

30.29 S. Thrun, M. Bennewitz, W. Burgard, A.B. Cremers, F. Dellaert, D. Fox, D. Haehnel, C. Rosenberg, N. Roy, J. Schulte, D. Schulz: MINERVA: A second geration mobile tour-guide robot, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (1999) pp. 1999–2005

30.30 K. Konolige: Improved occupancy grids for map building, Auton. Robotics **4**, 351–367 (1997)

30.31 R. Grabowski, P. Khosla, H. Choset: An enhanced occupancy map for exploration via pose separation, Proc. IEEE/RSJ Int. Conf. Intell. Robotics Syst. (IROS) (2003) pp. 705–710

30.32 J.D. Tardos, J. Neira, P.M. Newman, J.J. Leonard: Robust mapping and localization in indoor environments using sonar data, Int. J. Robotics Res. **21**(6), 311–330 (2002)

30.33 O. Aycard, P. Larouche, F. Charpillet: Mobile robot localization in dynamic environments using places recognition, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (1998) pp. 3135–3140

30.34 B. Kuipers, P. Beeson: Bootstrap learning for place recognition, Proc. 18th Nat. Conf. Artif. Intell. (ANAI) (2002)

30.35 A. Bandera, C. Urdiales, F. Sandoval: Autonomous global localization using Markov chains and optimized sonar landmarks, Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS) (2000) pp. 288–293

30.36 R. Kuc: Biomimetic sonar and neuromorphic processing eliminate reverberation artifacts, IEEE Sens. J. **7**(3), 361–369 (2007)

30.37 A.M. Sabatini: A stochastic model of the time-of-flight noise in airborne sonar ranging systems, IEEE Trans. Ultrason. Ferroelectr. Freq. Control **44**(3), 606–614 (1997)

30.38 C. Biber, S. Ellin, E. Sheck, J. Stempeck: The Polaroid ultrasonic ranging system, Proc. 67th Audio Eng. Soc. Conv. (1990)

30.39 R. Kuc: Forward model for sonar maps produced with the Polaroid ranging module, IEEE Trans. Robotics Autom. **19**(2), 358–362 (2003)

30.40 M.K. Brown: Feature extraction techniques for recognizing solid objects with an ultrasonic range sensor, IEEE J. Robotics Autom. **1**(4), 191–205 (1985)

30.41 N.L. Harper, P.J. McKerrow: Classification of plant species from CTFM ultrasonic range data using a neural network, Proc. IEEE Int. Conf. Neural Netw. (1995) pp. 2348–2352

30.42 Z. Politis, P.J. Probert: Target localization and identification using CTFM sonar imaging: The AURBIT method, Proc. IEEE Int. Symp. Comput. Intell. Robotics Autom. (CIRLA) (1999) pp. 256–261

30.43 R. Mueller, R. Kuc: Foliage echoes: A probe into the ecological acoustics of bat echolocation, J. Acoust. Soc. Am. **108**(2), 836–845 (2000)

30.44 P.N.T. Wells: *Biomedical Ultrasonics* (Academic, New York 1977)

30.45 J.L. Prince, J.M. Links: *Medical Imaging Signals and Systems* (Prentice Hall, Upper Saddle River 2006)

30.46 F.J. Alvarez, R. Kuc: High resolution adaptive spiking sonar, IEEE Trans. Ultrason. Ferroelectr. Freq. Control **56**(5), 1024–1033 (2009)

30.47 F.J. Alvarez, R. Kuc, T. Aguilera: Identifying fabrics with a variable emission airborne spiking sonar, IEEE Sens. J. **11**(9), 1905–1912 (2011)

30.48 J.J. Leonard, H.F. Durrant-Whyte: Mobile robot localization by tracking geometric beacons, IEEE Trans. Robotics Autom. **7**(3), 376–382 (1991)

30.49 R. Kuc: Generating B-scans of the environmental with conventional sonar, IEEE Sens. J. **8**(2), 151–160 (2008)

30.50 P.M. Woodward: *Probability and Information Theory with Applications to Radar*, 2nd edn. (Pergamon, Oxford 1964)

30.51 A. Heale, L. Kleeman: Fast target classification using sonar, Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS) (2001) pp. 1446–1451

30.52 S. Fazli, L. Kleeman: A real time advanced sonar ring with simultaneous firing, Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS) (2004) pp. 1872–1877

30.53 T. Yata, A. Ohya, S. Yuta: A fast and accurate sonar-ring sensor for a mobile robot, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (1999) pp. 630–636

30.54 L. Kleeman: Scanned monocular sonar and the doorway problem, Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS) (1996) pp. 96–103

30.55 G. Kao, P. Probert: Feature extraction from a broadband sonar sensor for mapping structured environments efficiently, Int. J. Robotics Res. **19**(10), 895–913 (2000)

30.56 B. Stanley, P. McKerrow: Measuring range and bearing with a binaural ultrasonic sensor, Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS) (1997) pp. 565–571

30.57 P.T. Gough, A. de Roos, M.J. Cusdin: Continuous transmission FM sonar with one octave bandwidth and no blind time. In: *Autonomous Robot Vehicles*,

Avidbots Ex1016
Page 815 of 2154

ed. by I.J. Cox, G.T. Wilfong (Springer, Berlin, Heidelberg 1990) pp. 117–122
30.58 L. Kay: A CTFM acoustic spatial sensing technology: Its use by blind persons and robots, Sens. Rev. **19**(3), 195–201 (1999)
30.59 L. Kay: Auditory perception and its relation to ultrasonic blind guidance aids, J. Br. Inst. Radio Eng. **24**, 309–319 (1962)
30.60 P.J. McKerrow, N.L. Harper: Recognizing leafy plants with in-air sonar, IEEE Sens. J. **1**(4), 245–255 (2001)
30.61 K. Audenaert, H. Peremans, Y. Kawahara, J. Van Campenhout: Accurate ranging of multiple objects using ultrasonic sensors, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (1992) pp. 1733–1738
30.62 J. Borenstein, Y. Koren: Noise rejection for ultrasonic sensors in mobile robot applications, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (1992) pp. 1727–1732
30.63 K.W. Jorg, M. Berg: Mobile robot sonar sensing with pseudo-random codes, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (1998) pp. 2807–2812
30.64 S. Shoval, J. Borenstein: Using coded signals to benefit from ultrasonic sensor crosstalk in mobile robot obstacle avoidance, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (2001) pp. 2879–2884
30.65 K. Nakahira, T. Kodama, T. Furuhashi, H. Maeda: Design of digital polarity correlators in a multiple-user sonar ranging system, IEEE Trans. Instrum. Meas. **54**(1), 305–310 (2005)
30.66 A. Heale, L. Kleeman: A sonar sensor with random double pulse coding, Aust. Conf. Robotics Autom. (2000) pp. 81–86
30.67 A. Diosi, G. Taylor, L. Kleeman: Interactive SLAM using Laser and Advanced Sonar, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (2005) pp. 1115–1120
30.68 S.A. Walter: The sonar ring: obstacle detection for a mobile robot, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (1987) pp. 1574–1578
30.69 S. Fazli, L. Kleeman: Wall following and obstacle avoidance results from a multi-DSP sonar ring on a mobile robot, Proc. IEEE Int. Conf. Mechatron. Autom. (2005) pp. 432–436
30.70 S. Fazli, L. Kleeman: Sensor design and signal processing for an advanced sonar ring, Robotica **24**(4), 433–446 (2006)
30.71 D. Browne, L. Kleeman: An advanced sonar ring design with 48 channels of continuous echo processing using matched filters, Proc. IEEE/RSJ Intell. Robots Syst. Conf. (IROS) (2009) pp. 4040–4046
30.72 D.C. Browne, L. Kleeman: A sonar ring with continuous matched filtering and dynamically switched templates, Robotica **30**(6), 891–912 (2012)
30.73 L. Kleeman, Akihisa Ohya: The design of a transmitter with a parabolic conical reflector for a sonar ring, Aust. Conf. Robotics Autom. (ICRA), Auckland (2006)
30.74 D.C. Browne, L. Kleeman: A double refresh rate sonar ring with FPGA-based continuous matched filtering, Robotica **30**(7), 1051–1062 (2012)
30.75 J. Steckel, A. Boen, H. Peremans: Broadband 3-D sonar system using a sparse array for indoor navigation, IEEE Trans. Robotics **91**, 1–11 (2012)
30.76 W.W.L. Au: *The Sonar of Dolphins* (Springer, Berlin, Heidelberg 1993)
30.77 R. Kuc, V. Kuc: Bat wing air pressures may deflect prey structures to provide echo cues for detecting prey in clutter, J. Acoust. Soc. Am. **132**(3), 1776–1779 (2012)
30.78 B. Barshan, R. Kuc: Bat-like sonar system strategies for mobile robots, Proc. IEEE Int. Conf. Syst. Man Cybern. (1991)
30.79 R. Kuc: Biologically motivated adaptive sonar, J. Acoust. Soc. Am. **100**(3), 1849–1854 (1996)
30.80 V.A. Walker, H. Peremans, J.C.T. Hallam: One tone, two ears, three dimensions: A robotic investigation of pinnae movements used by rhinolophid and hipposiderid bats, J. Acoust. Soc. Am. **104**, 569–579 (1998)
30.81 L. Gao, S. Balakrishnan, W. He, Z. Yan, R. Mueller: Ear deformations give bats a physical mechanism for fast adaptation of ultrasonic beam patterns, Phys. Rev. Lett. **1007**, 214–301 (2011)
30.82 R. Kuc: Biomimetic sonar system recognizes objects using binaural information, J. Acoust. Soc. Am. **102**(2), 689–696 (1997)
30.83 R. Kuc: Recognizing retro-reflectors with an obliquely-oriented multi-point sonar and acoustic flow, Int. J. Robotics Res. **22**(2), 129–145 (2003)
30.84 T. Horiuchi, T. Swindell, D. Sander, P. Abshire: A low-power CMOS neural amplifier with amplitude measurements for spike sorting, Proc. Int. Symp. Circuits Syst. (ISCAS), Vol. IV (2004) pp. 29–32
30.85 R. Kuc: Neuromorphic processing of moving sonar data for estimating passing range, IEEE Sens. J. **7**(5), 851–859 (2007)
30.86 R. Kuc: Binaural sonar electronic travel aid provides vibrotactile cues for landmark, reflector motion, and surface texture classification, IEEE Trans. Biomed. Eng. **49**(10), 1173–1180 (2002)

Avidbots Ex1016
Page 816 of 2154

Multimedia Contents

31. Range Sensing

Kurt Konolige, Andreas Nüchter

Range sensors are devices that capture the three-dimensional (3-D) structure of the world from the viewpoint of the sensor, usually measuring the depth to the nearest surfaces. These measurements could be at a single point, across a scanning plane, or a full image with depth measurements at every point. The benefits of this range data is that a robot can be relatively certain where the real world is, relative to the sensor, thus allowing the robot to more reliably find navigable routes, avoid obstacles, grasp objects, act on industrial parts, etc.

This chapter introduces the main representations for range data (point sets, triangulated surfaces, voxels), the main methods for extracting usable features from the range data (planes, lines, triangulated surfaces), the main sensors for acquiring it (Sect. 31.1 – stereo and laser triangulation and ranging systems), how multiple observations of the scene, for example, as if from a moving robot, can be registered (Sect. 31.3) and several indoor and outdoor robot applications where range data greatly simplifies the task (Sect. 31.4).

31.1 **Range Sensing Basics** 783
31.1.1 Range Images and Point Sets 783
31.2 **Sensor Technologies** 785
31.2.1 Triangulation 786
31.2.2 Time of Flight 786
31.2.3 Comparison of Methods 788
31.2.4 Stereo Vision 789
31.2.5 Structured Light Stereo 793
31.3 **Registration** 794
31.3.1 The ICP Algorithm 794
31.3.2 Marker and Feature-Based Registration 795
31.3.3 Multiple-View Registration 801
31.3.4 Model Matching 802
31.3.5 Relative Pose Estimation 802
31.3.6 3-D Applications 803
31.4 **Navigation and Terrain Classification and Mapping** 804
31.4.1 Mobile Mapping 804
31.4.2 Urban Navigation 805
31.4.3 Rough Terrain 806
31.5 **Conclusions and Further Reading** 807
References 807

31.1 Range Sensing Basics

Here we present the basic representations used for range image data and discuss some issues relating to representations and robotics applications. While there are ranging devices that use sound or other waves to determine distance, this chapter concentrates on sensors that use light.

31.1.1 Range Images and Point Sets

Range data is a two-and-a-half-dimensional (2.5-D) or 3-D representation of the scene around the robot. The 3-D aspect arises because we are measuring the (X, Y, Z) coordinates of one or more points in the scene. Often only a single range image is used at each time instance. This means that we only observe the front sides of objects – the portion of the scene visible from the robot. In other words, we do not have a full 3-D observation of all sides of a scene. This is the origin of the term 2.5-D. Figure 31.1 shows a sample range image and a registered reflectance image, where each pixel records the level of reflected infrared light.



Fig.31.1a,b 3-D laser scan with a field of view of 360° × 100° acquired in an urban environment (Bremen downtown). **(a)** Registered infrared reflectance image. **(b)** Range image where closer is darker

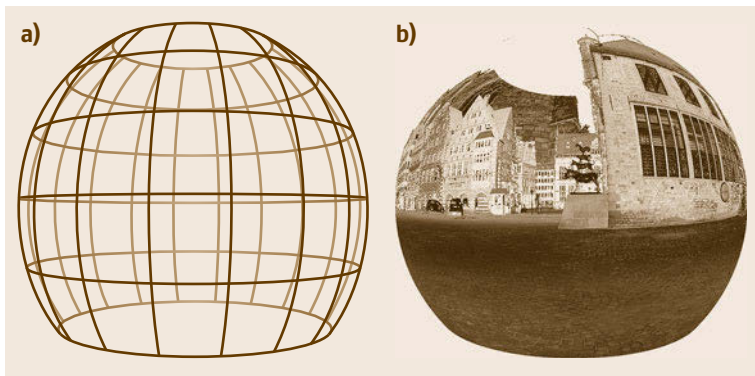


Fig. 31.2 **(a)** Mesh showing the rasterization of a 3-D range scanner using rotating mirrors and axis. **(b)** Sphere with superimposed reflectance values

There are two standard formats for representing range data. The first is an image $d(i,j)$, which records the distance d to the corresponding scene point (x,y,z) for each image pixel (i,j). There are several common mappings from $(i,j,d(i,j))$ to (X,Y,Z), usually arising from the geometry of the range sensor or from the application needs. The most common image mappings are illustrated in Fig. 31.3. Range scanners with rotating mirrors sample in spherical coordinates, that is,

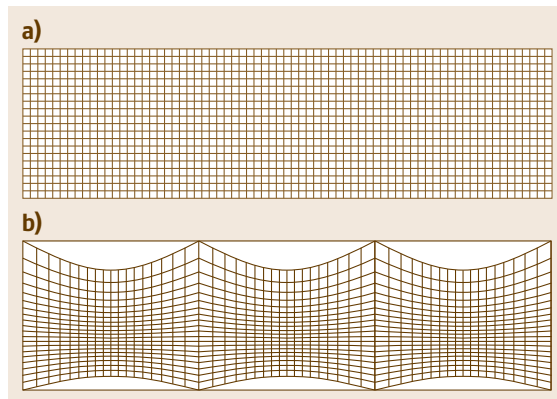


Fig. 31.3 **(a)** Equirectangular projection of a spherical range scan. Latidute and longitude mesh is shown. **(b)** Rectilinear projection with three images compined to unwrap a scanning sphere ►

Avidbots Ex1016
Page 818 of 2154



Fig. 31.4 3-D range scan represented with three rectilinear projections (Fig. 31.1)

(θ, φ, d). Figure 31.2 shows a sphere as longitude and latitude mesh and superimposed with the reflectance values of Fig. 31.1. The most natural range image representation is to put (θ, φ) on the (i, j)-axis of the range image, that is,

$$i = \theta \,,$$
$$j = \varphi \,,$$

where the longitude θ and the latitude φ are the spherical coordinates. This is a projection of the samples sphere to a two-dimensional (2-D) image, which includes a distortion. The conversion between $(i, j, d) = (\theta, \varphi, d)$ and (x, y, z) proceeds as

$$d = \sqrt{x^2 + y^2 + z^2} \,,$$
$$\theta = \arccos\left(\frac{z}{r}\right) \,,$$
$$\varphi = \arctan2\,(y, x) \,.$$

Conversely, the Cartesian coordinates may be retrieved from the spherical coordinates by

$$x = d \sin\theta \cos\varphi \,,$$
$$y = d \sin\theta \sin\varphi \,,$$
$$z = d \cos\theta \,.$$

The further common projection is the perspective one, usually called rectilinear projection. It is also called *gnomonic* or *tangent-plane* projection. This include range images produced by kinetic-like devices. Here the range values are assumed to be projected in a pinhole camera like fashion. The primary advantage of the rectilinear projection is that it maps straight lines in the 3-D space and in the 2-D image. Its disadvantage is the smaller field of view (FOV). The projection is stretched toward the corners and the distortion grow with larger fields of view. Furthermore, FOVs are often not realizable by the hardware. The projection proceeds as

$$i = \frac{\cos\varphi \sin(\theta - \theta_0)}{\sin\varphi_1 \sin\varphi + \cos\varphi_1 \cos\varphi \cos(\theta - \theta_0)} \,,$$
$$j = \frac{\cos\varphi_1 \sin\varphi - \sin\varphi_1 \cos\varphi \cos(\theta - \theta_0)}{\sin\varphi_1 \sin\varphi + \cos\varphi_1 \cos\varphi \cos(\theta - \theta_0)} \,,$$

where θ_0 and φ_1 are the central longitude and central latitude, respectively. Figure 31.4 shows a 360° scan composed of three rectilinear range images.

Further common projections are the isogonic Mercator projection, the cylindrical projection, Pannini projection, and the stereographic [31.1]. All these projections are used to unwrap the scanning sphere to a 2-D array representation. In some applications, orthographic projections are processed from 3-D point cloud data. Some range sensors only record distances in a slice, so the scene (x, z) is represented by the linear image $d(i)$ for each pixel i.

The second format is as a list $\{(x_i, y_i, z_i)\}$ of 3-D data points, but this format can be used with all of the mappings listed above. Given the conversions from image data $d(i, j)$ to (x, y, z) the range data is only supplied as a list, which is usually called the point cloud.

31.2 Sensor Technologies

There are two main technologies for range sensing: triangulation and time-of-flight. There are many variations of each of these, with different strengths and weaknesses. Here we briefly describe the basic concepts, and summarize the characteristics of the main types of sensors. More detailed descriptions of stereo and structured light sensors are given in subsequent sections.

Avidbots Ex1016
Page 819 of 2154

31.2.1 Triangulation

Triangulation sensors measure depth by determining the angle formed by the rays from a world point to two *sensors*. The sensors are separated by a baseline of length b, which forms the third segment of a triangle between the two sensors and the point. For simplicity, let one of the rays form a right angle with the baseline. Then the angle θ of the other sensor ray is related to the depth Z perpendicular to the baseline by

$$\tan\theta = \frac{Z}{b} . \tag{31.1}$$

An image sensor measures the angle θ by an offset on an image plane from the primary ray; this offset x is called the *disparity*. If we assume that the image plane is parallel to the baseline, then $\tan\theta = f/x$, and we get the basic equation of triangulation depth sensors

$$Z = \frac{fb}{x} . \tag{31.2}$$

Depth Precision

An important concept is the *depth resolution* of a sensor: how precisely can the sensor measure the depth? Differentiating with respect to x and substituting for x gives

$$\frac{\mathrm{d}Z}{\mathrm{d}x} = \frac{-Z^2}{fb} . \tag{31.3}$$

Triangulation precision falls off with the square of the distance to an object: if it is 1 mm at 1 m, then it is 4 mm at 2 m. Increasing the baseline or reducing the field of views FOV is inversely related to precision, that is, if the baseline is doubled or FOV is halved, the precision is halved (e.g., from 1 mm at 0.1 m baseline to 0.5 mm at 0.2 m baseline). This makes triangulation sensors more difficult to use at a distance; changing the FOV or baseline can help compensate, but brings in other tradeoffs (Sect. 31.2.4).

Types of Triangulation Sensors

The two main types are stereo cameras, which have two cameras separated by a baseline, and structured light sensors, which substitute a projector for one of the cameras.

Stereo cameras take images of a scene from two slightly different viewpoints, and match texture in the images to determine corresponding points and disparities. Some issues with stereo cameras are that there is often too little texture, especially indoors, to make reliable matches; and the matching uses small patches, which blurs out the spatial resolution. Many ingenious methods have been developed to deal with these issues, including painting a texture with a projector (not to be confused with structured light), using different kinds of support neighborhoods and regularization methods.

Structured light sensors project a known pattern from one end of a baseline, and view the pattern with a camera from the other. By decoding the pattern in the image, the placement of the pattern in the projector is determined, and hence the disparity. The simplest type of projectors output a point or a line, usually with a laser; the decoding problem is easy, but the amount of information is restricted to the point or line, and must be scanned to produce a full 3-D image.

A full-image structured light system projects a pattern covering the whole image, that can be decoded to get information at each pixel or small group of pixels. The PrimeSense technology, used in the Kinect, projects a known pattern of dots and decodes a small neighborhood to find the projected coordinates; thus this sensor also blurs out the spatial resolution. Other sensors project a time series of images with a vertical structure, so that each pixel on a horizontal line accumulates a unique code, which is then decoded against the projected images. Popular binary codes are Gray codes, which require N images if the image width is 2^N; and sinusoidal phase-shifted patterns, which typically use three images and are decoded by determining the phase at each point. These time-series structured light systems can produce very good spatial and depth resolution.

In general, structured light systems are not used outdoors because of interference from natural light, and for distant objects indoors. They can also have difficulty with high-contrast and specular objects, and the phase-based systems can suffer from 2π phase ambiguities. Relative motion between the sensor and the scene can also distort readings for time series and scanning sensors, although very fast projectors and cameras can alleviate this problem for the former.

31.2.2 Time of Flight

The principle behind time-of-flight (TOF) sensors is like that of radar: measure the time it takes for light to be projected out to an object and return. Because light travels about 0.3 m per nanosecond, very precise timers are required for direct TOF measurement; an alternative is indirect methods, typically phase difference with a modulated reference beam.

Because they measure time of flight, these sensors can theoretically have constant precision in depth measurement, no matter how far an object – unlike triangulation sensors, which fall off as the square of distance. But TOF sensors cannot duplicate the very fine precision of triangulation sensors for close objects,

Avidbots Ex1016
Page 820 of 2154

and hence are not used in close-range metrology applications, such as small object reconstruction or parts quality measurements.

Direct Time of Flight

In direct TOF sensors, the travel time is measured by a high-speed chronometer. Laser-based direct TOF range sensors are also called LIDAR (light detection and ranging) or LADAR (laser radar) sensors. The travel time multiplied by the speed of light (in the given medium – space, air, or water and adjusted for the density and temperature of the medium) gives the distance

$$2d = ct\,, \tag{31.4}$$

where d is the distance to the object, c is the speed of light, and t is the measured travel time. Error in measuring the time t gives rise to a proportional distance error. In practice, one tries to measure the peak of the output pulse, which has a finite extent; weak reflections from distant objects make it harder to measure this peak and hence the error tends to increase with distance. Averaging multiple readings can reduce the random error in these readings.

The simplest TOF sensors transmit only a single beam, thus range measurements are only obtained from a single surface point. Robotics applications usually need more information, so the range data is usually supplied as a vector of range to surfaces lying in a plane (Fig. 31.5) or as an image (Fig. 31.1). To obtain these denser representations, the laser beam is swept across the scene. Normally, the beam is swept by a set of mirrors rather than moving the laser and detector themselves (mirrors are lighter and less prone to motion damage). The most common technologies for this are using a stepper motor (for program-based range sensing) or rotating or oscillating mirrors for automatic scanning.

Scanning or averaging requires multiple TOF pulses at perhaps high repetition rates, which can give rise to ambiguity about which pulse is actually being received. If δt is the time between pulses, then the *ambiguity interval* for the device is $1/2c\delta t$, for example, for a pulse repetition rate of 100 kHz, the ambiguity interval is 1500 m. If an object is further than this, it will be seen as within the interval, with distance $z \bmod 1/2c\delta t$. An *unwinding algorithm* can recover the true depth, if range values for the scanning system change slowly, and start within the ambiguity interval.

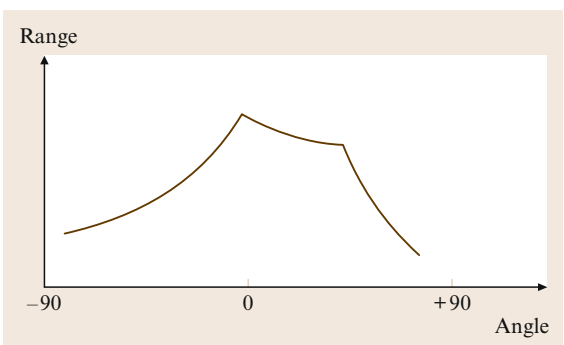


Fig. 31.5 Plot of ideal one-dimensional (1-D) range image of sample distance versus angle of measurement

Typical ground-based time of flight sensors suitable for robotics applications have a range of 10–100 m, and an accuracy of 5–10 mm. The amount of the scene scanned will depend on the sweep rate of the mirrors and the pulse rate, but 1–25k points per second are typical. Manufacturers of these sensors include Acuity, Hokuyo, Sick, Mensi, DeltaSphere, and Cyrax.

Multiple-beam scanning LIDARs can increase the amount of information available. Velodyne [31.2] makes three devices, with 16, 32, and 64 beams arranged vertically, that acquire point data at rates of up to 15 scans/s (1.3 MPixel/s), with a full 360° horizontal scan and a 27° vertical FOV from the laser array. The laser pulses are 5 ns long, and depth precision is approximately 2 cm. These devices are typically used in autonomous driving, for environment reconstruction and obstacle avoidance.

Flash LIDAR

In contrast to scanning devices, a *flash LIDAR* has a 2-D detector array (also called a *focal plane array* similar to a camera imager, but where each pixel incorporates timing circuitry to measure TOF of a laser pulse). Instead of a single or multiple laser beams, a light source pulse (LED (light-emitting diode) or laser) is shaped to cover a large area. All pixels have their timers started when the pulse is initiated, and measure the time it takes to receive the backscattered light. Typically, some tens of samples are captured and averaged, to reduce noise in the measurements – the amount of energy received is quite small, since the laser is not focused into a beam. As might be expected, the detector array pixels are quite large because of the timing electronics; a typical device from ASC [31.3] has 128×128 pixels, and can capture data at rates up to 60 Hz ($\approx$ 1 MPixel/s, similar to the Velodyne devices). These devices are expensive, and hence not used in consumer applications. Alternatively, indirect ways of measuring TOF have simpler per-pixel electronics – see the next subsection. Table 31.1 lists the characteristics of one of ASC's devices.

Table 31.1 Characteristics of TigerEye device

Manuf./Model	Resolution	Max range	Repeatability
ASC Inc. TigerEye	128×128	60–1100 m	0.04 m @ 60 m

Indirect Time of Flight Sensors

Indirect TOF sensors measure the distance by inferring transit times from particular properties of a propagating beam. The two main methods are based on *modulation and phase differences* and *gated intensity*.

Modulation-Based TOF

Modulation-based range sensors are commonly of two types, where a continuous laser signal is either amplitude or frequency modulated. By observing the phase shift between the outgoing and return signals, the signal transit time is estimated and from this the target distance.

Amplitude modulation works by varying the intensity of a light signal $s(t) = \sin(2\pi ft)$ with frequency f. The signal reflected from an object has a phase shift φ, and the returned signal is

$$r(t) = R\sin(2\pi ft - \varphi) = R\sin\left[2\pi f\left(t - \frac{2d}{c}\right)\right],$$
$$d = \frac{c\varphi}{4\pi f}, \tag{31.5}$$

with c is the speed of light, d the distance to the object, and R the reflected amplitude. Measuring the phase shift yields the distance; note because the phase shift wraps around at 2π, the ambiguity interval is $c/2f$. For a modulation of 10 MHz, the interval is 15 m.

To measure the phase difference, the returned signal is mixed with the original reference signal and a 90° shifted version, then low-pass filtered. Figure 31.6 compares the two mixed signals that give both the phase difference and the intensity.

There are commercial devices using single-beam and planar array versions of amplitude modulation. The planar arrays take advantage of CMOS integration to implement the full signal comparison at each pixel, in a compact sensor [31.4, 5]. Because the signal-to-noise ratio (SNR) depends on the amplitude of the returned signal, these devices typically have a depth resolution that falls off as d^2, comparable to triangulation devices. Typical repeatability is about 3 cm at 0.5 m, significantly less than triangulation devices; the arrays are up to 200×200 pixels. The attraction of the planar array is that it is a single-chip ranging device, simple and potentially cheap to produce. They also have good ambient light rejection, and potentially can be used outdoors.

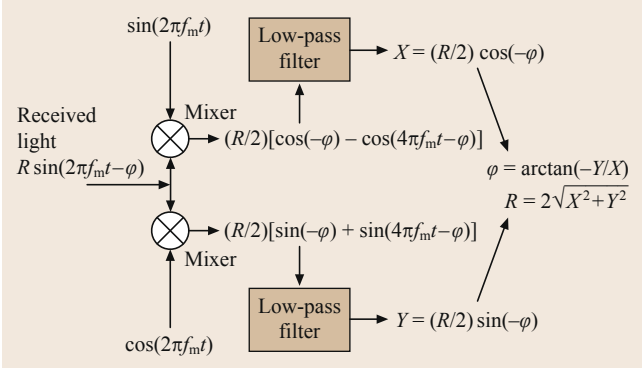


Fig. 31.6 Comparing the two mixed signals gives both the phase difference and the intensity

Frequency modulation, also called frequency modulation continuous wave (FMCW), modulates a laser frequency with a sawtooth ramp of period t_m, where the maximum frequency spread is Δf. The outgoing and incoming signals are mixed to give a signal difference f_i, which is measured by frequency counting. The distance is calculated as $d = f_i c t_m / 2\Delta f$. Since the frequency difference can be measured very accurately, this technique can yield excellent depth precision, on the order of 5 mm at 2 m [31.6]. The electronics required for mixing and frequency counting, as well as the difficulty of linear laser frequency modulation, have restricted this technique to high-end single-beam devices [31.7].

Range-Gated Intensity

Another indirect method for measuring TOF of a pulse is to measure the amount of the pulse that gets returned within a precise time period. If a pulse of width w is emitted, a detector will start to receive the pulse reflected from an object of distance d at $t_0 = 2d/c$, and finish receiving it at $t_w = (w + 2d)/c$. If the detector is *open* for a precise amount of time t, then the amount of the reflected signal it receives, relative to the full pulse reflection, is a linear measure of the distance d. Thus, by comparing two returns, one with a large open time to measure the full pulse, and one with a smaller open time, d can be determined. A planar array CMOS device can be fabricated to measure the two pulse returns sequentially, as long as each pixel can be triggered to be photosensitive for a precise amount of time. Commercial devices using this principle have precision on the order of a few cm at 1 m, with arrays up to 128×96 [31.8, 9].

31.2.3 Comparison of Methods

A useful overall summary of different types of range-sensing technology is in reference [31.6]. The two main parameters are spatial precision and depth precisions. In Fig. 31.7, available sensors are classified along these two dimensions. The direct TOF laser scanners (LMS, UT devices) cluster around 0.5–1 cm in both spatial and range precision. The Photon80 device uses FMCW

modulation with outstanding spatial and range precision; it is also a scanning device.

Triangulation devices (stereo, Kinect, and structured light in general) have less favorable characteristics. However, the target is at 2 m, and with triangulation precision falling with d^2, the comparison is biased. All these devices would be better than the TOF devices at 0.5 m in terms of range error. The stereo sensors can also exhibit much better spatial density in different configurations, that is, higher resolution sensors, larger baseline, smaller correlation window. In general, we would expect the triangulation sensors to outperform TOF sensors at shorter distances.

Finally, a single-chip flash LIDAR device, the IFM O3D200, uses amplitude modulation, and gives characteristically poor results relative to the other sensors.

31.2.4 Stereo Vision

This section discusses stereo analysis in more detail. Stereo analysis uses two or more input images to estimate the distance to points in a scene. The basic concept is *triangulation*: a scene point and the two camera points form a triangle, and knowing the baseline between the two cameras, and the angle formed by the camera rays, the distance to the object can be determined.

In practice, there are many difficulties in making a stereo imaging system that is useful for robotics applications. Most of these difficulties arise in finding reliable matches for pixels in the two images that correspond to the same point in the scene. A further consideration is that stereo analysis for robotics has a real-time constraint, and the processing power needed for some algorithms can be very high. But, in recent years much progress has been made, and the advantage of stereo imaging is that it can provide full 3-D range images, registered with visual information, potentially out to an infinite distance, at high frame rates – something which no other range sensor can match.

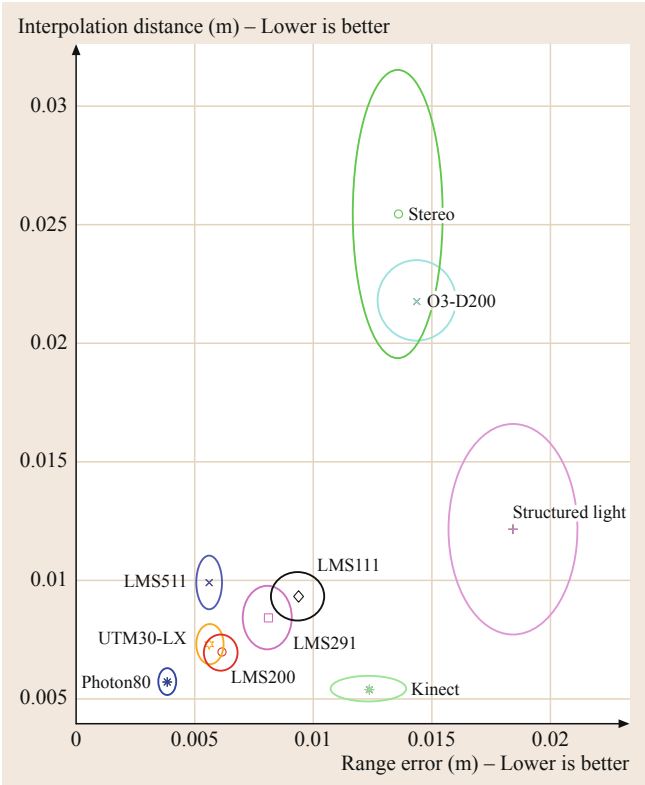


Fig. 31.7 Range device characteristics (after [31.6]). Spatial precision is on the Y-axis, depth precision on the X-axis. Measurements are taken on a target at 2 m

In this subsection, we will review the basic algorithms of stereo analysis, and highlight the problems and potential of the method. For simplicity, we use binocular stereo.

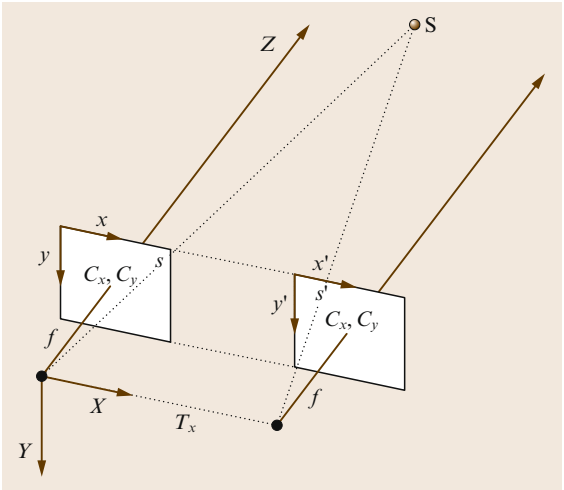


Fig. 31.8 Ideal stereo geometry. The global coordinate system is centered on the focal point (camera center) of the left camera. It is a right-handed system, with positive Z in front of the camera, and positive X to the right. The camera principal ray pierces the image plane at C_x, C_y, which is the same in both cameras (a variation for *verged* cameras allows C_x to differ between the images). The focal length is also the same. The images are lined up, with $y = y'$ for the coordinates of any scene point projected into the images. The difference between the x coordinates is called the *disparity*. The vector between the focal points is aligned with the X-axis ►

Avidbots Ex1016
Page 823 of 2154

Stereo Image Geometry

This subsection gives some more detail of the fundamental geometry of stereo, and in particular the relationship of the images to the 3-D world via projection and reprojection. A more in-depth discussion of the geometry and the rectification process can be found in [31.10].

The input images are *rectified*, which means that the original images are modified to correspond to ideal pinhole cameras with a particular geometry, illustrated in Fig. 31.8. Any 3-D point S projects to a point in the images along a ray through the focal point. If the principal rays of the cameras are parallel, and the images are embedded in a common plane and have collinear scan lines, then the search geometry takes a simple form. The *epipolar line* of a point s in the left image, defined as the possible positions of s′ in the right image, is always a scan line with the same y coordinate as s. Thus, search for a stereo match is linear. The process of finding a rectification of the original images that puts them into standard form is called *calibration*, and is discussed in [31.10].

The difference in the x coordinates of s and s′ is the *disparity* of the 3-D point, which is related to its distance from the focal point, and the baseline T_x that separates the focal points.

A 3-D point can be projected into either the left or right image by a matrix multiplication in homogenous coordinates, using the *projection matrix*. The 3-D coordinates are in the frame of the left camera (Fig 31.8)

$$\mathbf{P} = \begin{pmatrix} F_x & 0 & C_x & -F_x T_x \\ 0 & F_y & C_y & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} . \tag{31.6}$$

This is the projection matrix for a single camera. F_x and F_y are the focal lengths of the rectified images, and C_x and C_y are the optical center; T_x is the translation of the camera relative to the left (reference) camera. For the left camera, it is 0; for the right camera, it is the baseline times the x focal length.

A point in 3-D is represented by homogeneous coordinates and the projection is performed using a matrix multiply

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \mathbf{P} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} , \tag{31.7}$$

where $(x/w, y/w)$ are the idealized image coordinates.

If points in the left and right images correspond to the same scene feature, the depth of the feature can be calculated from the image coordinates using the *reprojection matrix*

$$\mathbf{Q} = \begin{pmatrix} 1 & 0 & 0 & -C_x \\ 0 & 1 & 0 & -C_y \\ 0 & 0 & 0 & F_x \\ 0 & 0 & -1/T_x & \frac{(C_x - C_{x'})}{T_x} \end{pmatrix} . \tag{31.8}$$

The primed parameters are from the left projection matrix, the unprimed from the right. The last term is zero except for verged cameras. If x, y and x', y are the two matched image points, with $d = x - x'$, then

$$\begin{pmatrix} X \\ Y \\ Z \\ W \end{pmatrix} = \mathbf{Q} \begin{pmatrix} x \\ y \\ d \\ 1 \end{pmatrix} , \tag{31.9}$$

where $(X/W, Y/W, Z/W)$ are the coordinates of the scene feature, and $d = x - x'$ is the disparity. Assuming $C_x = C'_x$, the Z distance assumes the familiar inverse form of triangulation

$$Z = \frac{F_x T'_x}{d} . \tag{31.10}$$

Reprojection is valid only for rectified images – for the general case, the projected lines do not intersect. The disparity d is an *inverse depth* measure, and the vector (x, y, d) is a perspective representation range image (Sect. 31.1.1), sometimes called the *disparity space* representation. The disparity space is often used in applications instead of 3-D space, as a more efficient representation for determining obstacles or other features (Sect. 31.4.3).

Equation (31.9) is a homography between disparity space and 3-D Euclidean space. Disparity space is also useful in translating between 3-D frames. Let $p_0 = [x_0, y_0, d_0, 1]$ in frame 0, with frame 1 related by the rigid motion R, t. From the reprojection (31.9) the 3-D position is $\mathbf{Q}p_0$. Under the rigid motion, this becomes

$$\begin{pmatrix} R & t \\ 0 & 1 \end{pmatrix} \mathbf{Q} p_0 ,$$

and finally applying $\mathbf{Q}^{-1}$ yields the disparity representation in frame 1. The concatenation of these operations is the homography

$$H(R, t) = \mathbf{Q}^{-1} \begin{pmatrix} R & t \\ 0 & 1 \end{pmatrix} \mathbf{Q} . \tag{31.11}$$

Using the homography allows the points in the reference frame to be directly projected onto another frame, without translating to 3-D points.

Stereo Matching Methods

The fundamental problem in stereo analysis is matching image elements that represent the same object or object part in the scene. Once the match is made, the range to the object can be computed using the image geometry.

Matching methods can be characterized as local or global. Local methods attempt to match small regions of one image to another based on intrinsic features of the region. Global methods supplement local methods by considering physical constraints such as surface continuity or base of support. Local methods can be further classified by whether they match discrete features among images, or correlate a small area patch [31.11]. Features are usually chosen to be lighting and viewpoint independent, for example, corners are a natural feature to use because they remain corners in almost all projections. Feature-based algorithms compensate for viewpoint changes and camera differences, and can produce rapid, robust matching. But they have the disadvantage of requiring perhaps expensive feature extraction, and yielding only sparse range results.

In the next section, we present local area correlation in more detail, since it is one of the most efficient and practical algorithms for real-time stereo. A survey and results of recent stereo matching methods are in [31.12], and the authors maintain a web page listing up-to-date information in [31.13].

Area Correlation Stereo

Area correlation compares small patches among images using correlation. The area size is a compromise, since small areas are more likely to be similar in images with different viewpoints, while larger areas increase the signal-to-noise ratio. In contrast to the feature-based method, area-based correlation produces dense results. Because area methods need not compute features, and have an extremely regular algorithmic structure, they can have optimized implementations.

The typical area correlation method has five steps (Fig. 31.9):

1. Geometry correction. In this step, distortions in the input images are corrected by warping into a *standard form*.
2. Image transform. A local operator transforms each pixel in the grayscale image into a more appropriate form, for example, normalizes it based on average local intensity.
3. Area correlation. This is the correlation step, where each small area is compared with other areas in its search window.
4. Extrema extraction. The extreme value of the correlation at each pixel is determined, yielding a disparity image: each pixel value is the disparity between the left and right image patches at the best match.
5. Post-filtering. One or more filters clean up noise in the disparity image result.

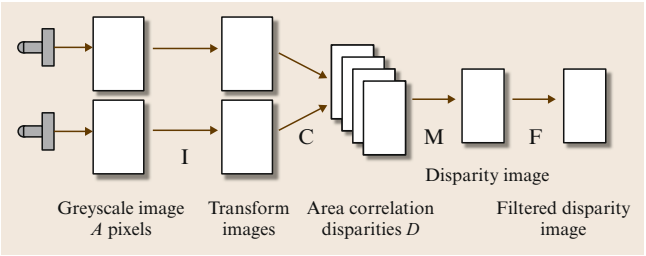


Fig. 31.9 Basic stereo processing (see text for details)

Correlation of image areas is disturbed by illumination, perspective, and imaging differences among images. Area correlation methods usually attempt to compensate by correlating not the raw intensity images, but some transform of the intensities. Let u, v be the center pixel of the correlation, d the disparity, and $I_{x,y}$, $I'_{x,y}$ the intensities of the left and right images:

1. Normalized cross-correlation

$$\frac{\sum_{x,y}[I_{x,y} - \hat{I}_{x,y}][I'_{x-d,y} - \hat{I'}_{x-d,y}]}{\sqrt{\sum_{x,y}[I_{x,y} - \hat{I}_{x,y}]^2 \sum_{x,y}[I'_{x,y} - \hat{I'}_{x-d,y}]^2}}$$

2. High-pass filter such as Laplacian of Gaussian (LOG). The Laplacian measures directed edge intensities over some area smoothed by the Gaussian. Typically, the standard deviation of the Gaussian is 1–2 pixels

$$\sum_{x,y} s(\mathrm{LOG}_{x,y} - \mathrm{LOG}_{x-d,y}) \ ,$$

where $s(x)$ is x^2 or $||x||$.

3. Nonparametric. These transforms are an attempt to deal with the problem of outliers, which tend to overwhelm the correlation measure, especially using a square difference. The census method [31.14] computes a bit vector describing the local environment of a pixel, and the correlation measure is the Hamming distance between two vectors

$$\sum_{x,y} (I_{x,y} > I_{u,v}) \oplus \left(I'_{x-d,y} > I'_{u,v}\right) \ .$$

Results on the different transforms and their error rates for some standard images are compiled in [31.13].

Another technique for increasing the signal-to-noise ratio of matches is to use more than two images [31.15].

Avidbots Ex1016
Page 825 of 2154

Table 31.2 Post-filtering techniques for eliminating false matches in area correlation

Correlation surface [31.17]	Peak width: Wide peak indicates poor feature localization Peak height: Small peak indicates poor match Number of peaks: Multiple peaks indicate ambiguity
Mode filter	Lack of supporting disparities violates smoothness
Left/right check [31.18, 19]	Nonsymmetric match indicates occlusion
Texture [31.20]	Low texture energy yields poor matches

This technique can also overcome the problem of viewpoint occlusion, where the matching part of an object does not appear in the other image. The simple technique of adding the correlations between images at the same disparity seems to work well [31.16]. Obviously, the computational expenditure for multiple images is greater than that for two.

Dense range images usually contain false matches that must be filtered, although this is less of a problem with multiple-image methods. Table 31.2 lists some of the post-filters that have been discussed in the literature.

Disparity images can be processed to give subpixel accuracy, by trying to locate the correlation peak between pixels. This increases the available range resolution without much additional work. Typical accuracies are 1/10 pixel.

Stereo Range Quality

Various artifacts and problems affect stereo range images.

Smearing. Area correlation introduces expansion in foreground objects, for example, the woman's head in Fig. 31.10. The cause is the dominance of strong edges on the object. Nonparametric measures are less subject to this phenomenon. Other approaches include multiple correlation windows and shaped windows.

Dropouts. These are areas where no good matches can be found because of low texture energy. Dropouts are a problem for indoor and outdoor man-made surfaces. Projecting a random texture can help [31.21].

Range Resolution. Unlike LADAR devices, stereo range accuracy is a quadratic function of distance, found by differentiating (31.10) with respect to disparity

$$\delta Z = -\frac{F_x T'_x}{d^2} \,. \tag{31.12}$$

The degradation of the stereo range with distance can be clearly seen in the 3-D reconstruction of Fig. 31.10.

Processing. Area correlation is processor-intensive, requiring Awd operations, where A is the image area, w is the correlation window size, and d is the number of disparities. Clever optimizations take advantage of redundant calculations to reduce this to Ad (independent of window size), at the expense of some storage. Real-time implementations exist for standard personal computers (PC)s [31.22, 23], graphics accelerators [31.24, 25], digital signal processors (DSPs) [31.26], field programmable gate arrays (FPGAs) [31.22, 27], and specialized application-specific integrated circuits (ASICs) [31.28].

Other Visual Sources of Range Information

Here we briefly list the most popular, but less reliable sources of range information. These sources can potentially supplement other sensors:

- *Focus/defocus*: Knowledge of the camera parameters and the amount of blur of image features allows the estimation of how far the corresponding scene features are from the perfect focus distance [31.29]. Sensors may be passive (using a precaptured image) or active (capturing several images with different focus settings).

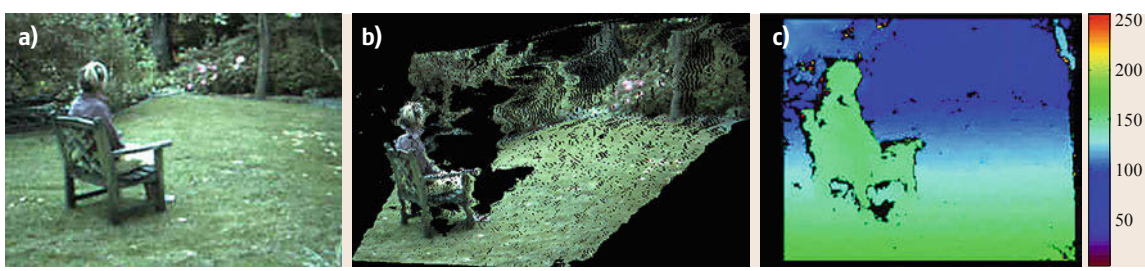


Fig.31.10a–c Sample stereo results from an outdoor garden scene; baseline is 9 cm. **(a)** original left image. **(b)** computed 3-D points from a different angle. **(c)** disparity in pseudo-color

- *Structure and motion*: Structure and motion algorithms compute 3-D scene structure and the sensor positions simultaneously [31.30]. This is essentially a binocular stereo process (see discussion before), except that only a single moving camera is used. Thus, the images needed by the stereo process are acquired by the same camera in several different positions. Video camcorders are also used, but they have lower resolution. One important advantage of this approach over the normal algorithm is that features can be tracked easily if the time between frames or the motion is small enough. This simplifies the *correspondence problem*; however, it can lead to another problem. If the pair of images used for the stereo calculation are taken close together in time, then the separation between the cameras' images will not be much – this is a *short baseline*. Triangulation calculations are then more inaccurate, as small errors in estimating the position of image features results in large errors in the estimated 3-D position (particularly the depth estimate). This problem can be partly avoided by tracking for longer periods. A second problem that can arise is that not all motions are suitable for estimate of the full 3-D scene structure. For example, if the video recorder only rotates about its optical axis or about its focus point, then no 3-D information can be recovered. A little care can avoid this problem.
- *Shading*: The pattern of shading on a surface is related to the orientation of the surface relative to the observer and light sources. This relationship can be used to estimate the surface orientation across the surface. The surface normals can then be integrated to give an estimate of the relative surface depth.
- *Photometric stereo*: Photometric stereo [31.31] is a combination of shading and stereo processes. The key concept is that the shading of an object varies with the position of the light sources. Hence, if you had several aligned pictures of an object or scene with the light source in different positions (e.g., the sun moved), then you can calculate the scene's surface normals. From these, the relative surface depth can be estimated. The restriction of a stationary observer and changing light sources makes this approach less likely to be useful to most robotics applications.
- *Texture*: The way uniform or statistical textures vary on a surface is related to the orientation of the surface relative to the observer. As with shading, the texture gradients can be used to estimate the surface orientation across the surface [31.32]. The surface normals can then be integrated to give an estimate of the relative surface depth.

31.2.5 Structured Light Stereo

Recently, the Kinect range-sensing device became the fastest-selling consumer electronics device in history [31.33]. The Kinect uses a structured-light stereo technique from PrimeSense (bought out by Apple, Inc.) to give a reasonable balance of spatial and depth precision, in a highly-integrated device, at a very low price.

The PrimeSense technology projects a structured pattern generated from an IR (infrared) laser and patented combination of diffractive elements (Fig. 31.11). The pattern consists of bright dots on a grid, with unique combinations of dots over a horizontal area. A camera with resolution 1280×960 captures the IR image at a horizontal offset of 7.5 cm from the projector, and correlates a 19×19 block against the known dot pattern for a given horizontal line. If a match is found, the position of the pattern in the known projected pattern gives the angle subtended by the reflecting object to the projector and camera, and hence the distance. A depth image is returned at a resolution of 640×480, with 11 bits of depth precision, over an FOV of approximately 50°; the depth is interpolated to a precision of 1/8 pixel. Error characteristics for the Kinect have been studied in [31.34]. The depth decision is limited by the resolution, similarly to (31.3).

There are additional sources of error. *Dropouts* occur when the surfaces are minimally reflective, or specular. *False positives*, where there are gross mismatches in the match, are very rare with PrimeSense technology. *Spatial precision* is much lower than the returned 640×480 depth image would indicate, because of the *smear-*



Fig. 31.11 The laser grid of the Kinect for calculating depth

ing effect of block correlation. The projected pattern expands and contracts based on temperature, leading to offset in disparities along the vertical axis. While the projector is temperature controlled via a Peltier unit, its temperature can vary. A temperature sensor measures this variation, and applies disparity steps across the image, which are apparent as vertical steps in the depth image.

31.3 Registration

This section introduces techniques for 3-D localization of parts for robot manipulation, self-localization of robot vehicles, and scene understanding for robot navigation. All of these are based on the ability to register 3-D shapes, for example, range images to range images, triangulated surfaces or geometric models. Registration puts two or more independently acquired 3-D shapes into one frame of reference. It can be solved as an optimization problem that uses a cost function for the quality of the alignment. The 3-D shapes are registered by determining the rigid transformation (rotation and translation) which minimizes the cost function. Feature based registration extracts distinguishing features of the range images and uses corresponding features for calculating the alignment.

31.3.1 The ICP Algorithm

The following method is used for the registration of point sets. The complete algorithm was invented at the same time in 1991 by *Besl* and *McKay* [31.35], by *Chen* and *Medioni* [31.36], and by *Zhang* [31.37]. The method is called iterative closest points (ICP) algorithm. It is the de-facto standard for registration of point sets, but is also applicable to 3-D shapes, if one samples them.

Given two independently acquired sets of 3-D points, $\hat{M}$ (model set) and $\hat{D}$ (data set) which correspond to a single shape, we want to find the transformation $(\mathbf{R}, \boldsymbol{t})$ consisting of a rotation matrix $\mathbf{R}$ and a translation vector $\boldsymbol{t}$ which minimizes the following cost function,

$$E(\mathbf{R}, \boldsymbol{t}) = \frac{1}{N} \sum_{i=1}^{N} \|\boldsymbol{m}_i - (\mathbf{R}\boldsymbol{d}_i + \boldsymbol{t})\|^2 . \qquad (31.13)$$

All the corresponding points can be represented in a tuple $(\boldsymbol{m}_i, \boldsymbol{d}_i)$ where $\boldsymbol{m}_i \in M \subset \hat{M}$ and $\boldsymbol{d}_i \in D \subset \hat{D}$. Two things have to be calculated. First, the corresponding points, and second, the transformation $(\mathbf{R}, \boldsymbol{t})$ that minimizes $E(\mathbf{R}, \boldsymbol{t})$ on the basis of the corresponding points. The ICP algorithm uses closest points as corresponding points. A sufficiently good starting guess enables the ICP algorithm to converge to the correct minimum. Figure 31.12 shows two 3-D point clouds, their initial alignment and the final registration after a few ICP iterations.

Implementations of ICP (Algorithm 31.1) use a maximal distance for closest points to handle partially overlapping point sets. In this case, the proof about ICP's monotonic convergence in [31.35] no longer holds, since the number of points as well as the value of $E(\mathbf{R}, \boldsymbol{t})$ might increase after applying a transformation.

Algorithm 31.1 The ICP algorithm

1: **for** $i = 0$ to *maxIterations* **do**
2: **for all** $\boldsymbol{d}_j \in D$ **do**
3: find the closest point within a range $d_{\max}$ in the set M for point $\boldsymbol{d}_j$
4: **end for**
5: Calculate transformation $(\mathbf{R}, \boldsymbol{t})$ that minimizes the error function (31.13)
6: Apply the transformation found in step 5 to the data set D.
7: Compute the difference of the quadratic error, that is, compute the difference of the value $\|E_{i-1}(\mathbf{R}, \boldsymbol{t}) - E_i(\mathbf{R}, \boldsymbol{t})\|$ before and after the ap-

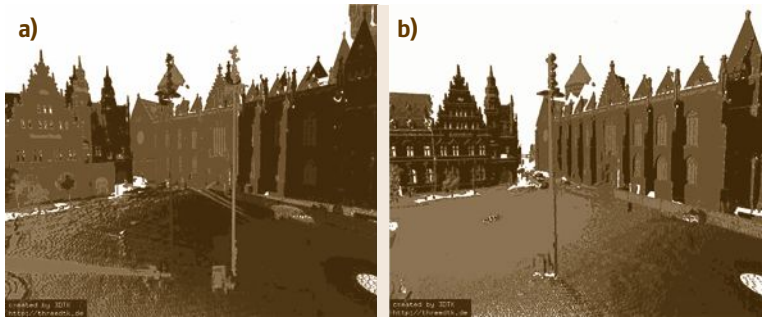


Fig.31.12a,b Initial alignment of two 3-D point clouds **(a)** and after optimization with ICP **(b)**

Avidbots Ex1016
Page 828 of 2154

plication of the transformation. If this difference falls below a threshold ε, terminate.

8: **end for**

Current research in the context of ICP algorithms mainly focuses on fast variants of ICP algorithms [31.38]. If the input are 3-D meshes then a point-to-plane metric can be used instead of (31.13). Minimizing using a point-to-plane metric outperforms the standard point-to-point one, but requires the computation of normals and meshes in a preprocessing step.

The computation of closest points is the crucial step of the ICP algorithm. A naive implementation examines all points in $\hat{M}$ and the resulting computation time for ICP is $O(|\hat{D}|\,|\hat{M}|)$, that is, $O(n^2)$. Note that $\hat{N}$ can be very large; advanced high-precise 3-D laser scanners such as the Zoller + Fröhlich yield a data rate up to 1 000 000 3-D points per second. An efficient tree data structure, a k-d tree, is widely used to speed up the closest point computation. Every node of the k-d tree represents a partition of the point set into two distinct sets, the successor nodes. The root of the tree represents the whole point set. The leaves of the tree are called buckets and are a partition of the set into small, disjunctive point sets. Furthermore, every node of the tree consists of the center and the dimension of the point set. In the original k-d tree paper, these so-called split dimensions have been chosen depending on the depth of node in a robin-round fashion [31.39]. The split dimension and value define an axis-aligned hyperplane in the k-dimensional space. The data are partitioned according to their position to the hyperplane into the successor nodes. The k-d tree is constructed until the number of points in the nodes falls below a threshold b (bucket size). Only the leaves of the tree contain the data points. Searching in k-d trees is done recursively. A given 3-D point $\boldsymbol{p}_q$ needs to be compared to the separating plane (splitting dimension and splitting value) in order to decide on which side the search must continue. This procedure is executed until the leaves are reached. There, the algorithm has to evaluate all bucket points. However, the closest point may be in a different bucket, iff the distance d of the query point $\boldsymbol{p}_\mathrm{q}$ to the limits is smaller than the one to the closest point in the bucket $\boldsymbol{p}_\mathrm{b}$. In this case backtracking has to be performed. The test is known as Ball-Within-Bounds test [31.39–41]. The optimized k-d tree chooses the split dimension and split value [31.40], such that the expected amount of backtracking is minimized. Since one typically has no information about the query points k-d tree algorithms take only the distribution of the given points into account. For all possible queries, this works sufficiently, but it will not be optimal for a specific query [31.40]. This enables the recursive construction and avoids the overall optimization that is known to be NP-complete [31.42].

Improvements to k-d tree search, especially for small dimensions, have been shown in the last decade. They include approximate k-d tree search [31.41], registration using d^2-trees [31.43] and cached k-d tree search [31.44]. In addition, the spatial data structure octree might, which is discussed later in this section, can be used to search the point clouds with similar performance.

In each ICP iteration, the transformation can be calculated in $O(N)$ by any of these four methods:

1. A singular value decomposition (SVD)-based method by *Arun* et al. [31.45]
2. A quaternion-based method by *Horn* et al. [31.46]
3. An algorithm using orthonormal matrices by *Horn* [31.47]
4. A calculation based on dual quaternions by *Walker* et al. [31.48]. Besides these closed-form solutions, there are several linearized, approximative version [31.49].

The challenge is to ensure thar $\mathbf{R}$ is orthonormal. Most often, the first method is implemented, due to it's simplicity and the availability of numerical SVDs in various libraries. The rotation $\mathbf{R}$ is represented as an orthonormal 3×3 matrix. The optimal rotation is calculated by $\mathbf{R}=\mathbf{V}\mathbf{U}^\mathrm{T}$. Here the matrices $\mathbf{V}$ and $\mathbf{U}$ are derived by the SVD $\mathbf{H}=\mathbf{U}\mathbf{\Lambda}\mathbf{V}^\mathrm{T}$ of a cross-correlation matrix $\mathbf{H}$. This 3×3 matrix $\mathbf{H}$ is given by

$$\mathbf{H}=\sum_{i=1}^{N}\boldsymbol{m}_i'^\mathrm{T}\boldsymbol{d}_i'=\begin{pmatrix}S_{xx}&S_{xy}&S_{xz}\\S_{yx}&S_{yy}&S_{yz}\\S_{zx}&S_{zy}&S_{zz}\end{pmatrix}, \tag{31.14}$$

where

$$S_{xx}=\sum_{i=1}^{N}m'_{x,i}d'_{x,i},\ S_{xy}=\sum_{i=1}^{N}m'_{x,i}d'_{y,i},\ \ldots\ .$$

31.3.2 Marker and Feature-Based Registration

To avoid issues with starting guess in the ICP framework, marker-based registration uses defined artificial or natural landmarks as corresponding points. This manual data association ensures that by minimizing (31.13) the scans are registered at the correct location. Iterations are no longer required, but possible as verification with the RANSAC algorithm. The RANSAC algorithm is a general, randomized procedure that iter-

atively finds an accurate model for observed data that may contain a large number of outliers [31.50].

3-D Feature Representations and Extraction

There are many representations available for encoding 3-D scene structure and model representations, but the following representations are the ones most commonly encountered in robotics applications. Some scene models or descriptions may use more than one of these simultaneously to describe different aspects of the scene or object models.

Normals. Normals describe the surface orientation in a points. There are a number of approaches that have been developed for computing normals in point clouds and range images. Most of these approaches involves some form of eigenvalue decomposition resembling total linear least squares. The normals are usually computed when taking into account the nearby neighbor points which are computed using a variety of methods like nearest neighbor search. These methods include k nearest neighbors (k-NN) and radius search among others. The method of total least squares is robust to noise as it inherently includes low-pass filtering, but it is sensitive to the distribution and density of the point samples as well as the curvature of the underlying manifold. Improvements to the above method by using higher order surfaces are discussed in [31.51] and it is noted that such methods can fail even with arbitrarily dense samples.

The total least-squares problem finds the plane parameters that optimally fit a small surface area in a point set given by the the nearest neighbors. A plane is defined by

$$n_x x + n_y y + n_z z - d = 0 \,,$$

where $\boldsymbol{p} = (x, y, z)^{\mathrm{T}}$ lies on the plane and (n_x, n_y, n_z, d) are the parameters to compute. Given a subset of k 3-D points $p_i, i = 1, 2, \ldots, k$ of the surface, least squares finds the optimal normal vector $n = (n_x, n_y, n_z)^{\mathrm{T}}$ and scalar d that minimizes the following error equation

$$e = \sum_{i=1}^{k} (\boldsymbol{p}_i \boldsymbol{n} - d) \,. \tag{31.15}$$

The basic method for normal estimation from the neighboring points using a fitting plane is the principal component analysis (PCA). PCA is the eigenvalue decomposition of the data covariance (or correlation) matrix or singular value decomposition of a data matrix, usually after mean centering (and normalizing) the data matrix for each attribute (3-D query point). PCA can be interpreted as fitting a Gaussian distribution and computing the main axes. For each query point

$$\boldsymbol{\mu} = \frac{1}{k} \sum_{i=1}^{k} \boldsymbol{p}_i \,,$$

$$\boldsymbol{\Sigma} = \frac{1}{k} \sum_{i=1}^{k} (\boldsymbol{p}_i - \boldsymbol{\mu})^{\mathrm{T}} (\boldsymbol{p}_i - \boldsymbol{\mu}) \,.$$

The vector corresponding to the smallest eigenvalue of the above matrix $\boldsymbol{\Sigma}$ is the normal direction if the neighboring points belong to a plane and this is also the closed form solution of the total least-squares problem.

For equirectangular range images, that is, spherical coordinates, a fast algorithm has been developed in [31.52] that avoids computing eigenvalues. Dividing (31.15) by d^2 gives a simplified function and further division by the squared range ρ^2 yields

$$e = \sum_{i=1}^{k} \left((\rho^{-1} \boldsymbol{p}_i)^{\mathrm{T}} \boldsymbol{n} - \rho_i^{-1} \right)^2 \,,$$

$$\boldsymbol{p}_i = \begin{pmatrix} \cos\theta_i \sin\varphi_i \\ \sin\theta_i \sin\varphi_i \\ \cos\varphi_i \end{pmatrix} .$$

Therefore, a solution for $\boldsymbol{n}$ is given as

$$\boldsymbol{n} = \mathbf{M}^{-1} \boldsymbol{b} \,,$$

where

$$\mathbf{M} = \sum_{i=1}^{k} \boldsymbol{p}_i \boldsymbol{p}_i^{\mathrm{T}} \,, \boldsymbol{b} = \sum_{i=1}^{k} \frac{\boldsymbol{p}_i}{\rho_i} \,.$$

This way the computation of eigenvalues is avoided and the matrix $\mathbf{M}$ can be precomputed for the desired image coordinates as it does not depend on the range. The tangential surface and therefore its normal vector are obtained by simply taking the derivative of the surface function in the point of interest

$$\boldsymbol{n} = \nabla\rho = \nabla\rho(\theta, \varphi)$$

$$= \begin{pmatrix} \cos\theta \sin\theta - \frac{\sin\theta}{\rho \sin\varphi} \frac{\partial\rho}{\partial\theta} + \frac{\cos\theta \cos\varphi}{\rho} \frac{\partial\rho}{\partial\varphi} \\ \sin\theta \sin\theta - \frac{\cos\theta}{\rho \sin\varphi} \frac{\partial\rho}{\partial\theta} + \frac{\sin\theta \cos\varphi}{\rho} \frac{\partial\rho}{\partial\varphi} \\ \cos\varphi - \frac{\sin\varphi}{\rho} \frac{\partial\rho}{\partial\varphi} \end{pmatrix} .$$

3-D Point Features. This is a set $\{\boldsymbol{p}_i = (x_i, y_i, z_i)\}$ of 3-D points that describe some salient and identifiable points in the scene. They might be the centers of spheres (often used as markers), corners where three planes

Avidbots Ex1016
Page 830 of 2154

intersect, or the extrema of some protrusion or indentation on a surface. They may be a subset of an initially acquired 3-D full scene point set, or they might be extracted from a range image, or they might be computed theoretical points based on extracted data features.

An early example of 3-D point features and descriptors are spin images, which are used for surface matching between point clouds and meshes. Scanned 3-D points become the vertices of the mesh and connectivity is established by the 3-D scanning geometry. A fundamental component of the surface matching representation is an oriented point, a three-dimensional point with an associated direction. *Huber* et al. define an oriented point $\boldsymbol{O}$ at a surface mesh vertex using the 3-D position of the vertex $\boldsymbol{p}$ and the surface normal at the vertex $\boldsymbol{n}$ [31.53]. Two coordinates can be calculated given an oriented point: α the radial distance to the surface normal line L and β the axial distance above the tangent plane P (Fig 31.13).

$$(\boldsymbol{x}) \mapsto (\alpha, \beta)$$
$$= \left(\sqrt{||\boldsymbol{x}-\boldsymbol{p}||^2 - [\boldsymbol{n}\cdot(\boldsymbol{x}-\boldsymbol{p})]^2}\ .\boldsymbol{n}\cdot(\boldsymbol{x}-\boldsymbol{p})\right)$$

The term spin map originates from the cylindrical symmetry of the oriented point basis; the basis can spin about its axis with no effect on the coordinates of points with respect to the basis [31.53]. A consequence of the cylindrical symmetry is that points that lie on a circle that is parallel to P and centered on L will have the same coordinates (α, β) with respect to the basis.

Seminal work on 3-D point features are the point feature histogram (PFH) descriptors. They encode a point's k-neighborhood geometrical properties by generalizing the mean curvature around the point using a multidimensional histogram of values [31.54]. This highly dimensional hyperspace aims at providing an informative signature for the feature representation, being invariant to the six-dimensional (6-D) pose of the underlying surface, and coping very well with different sampling densities or noise levels present in the neighborhood [31.55].

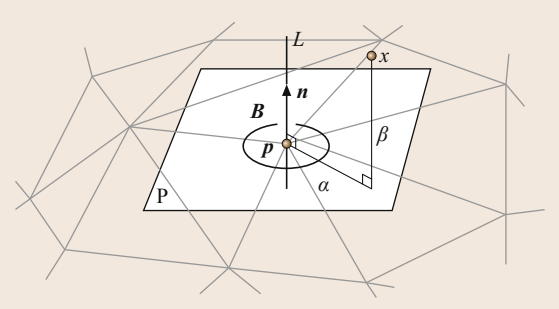


Fig. 31.13 Definition of a spin image

To formulate the new feature space, the concept of a dual-ring neighborhood is first introduced. Following the notation and the text of [31.54], let P be a set of 3-D points with x_i, y_i, z_i being the geometric coordinates. A point $\boldsymbol{p}_i$ of P is said to have a dual-ring neighborhood if

$$r_1, r_2 \in R r_1 < r_2\ ,\ \text{such that}\ \begin{cases} r_1 \to P^{k_1} \\ r_2 \to P^{k_2} \end{cases}\ ,$$

with $0 < k_1 < k_2$. The two radii r_1 and r_2 are used to determine two distinct layers of feature representations for $\boldsymbol{p}_i$. The first layer represents the surface normal at the query point from the neighborhood patch P^{k_1}. The second layer comprises the PFH as a set of angular features (Fig. 31.14)

$$\alpha = \boldsymbol{v}\cdot\boldsymbol{n}_\mathrm{t}\ ,$$
$$\varphi = \boldsymbol{u}\cdot\frac{(\boldsymbol{p}_\mathrm{t}-\boldsymbol{p}_\mathrm{s})}{d}\ ,$$
$$\theta = \arctan(\boldsymbol{w}\cdot\boldsymbol{n}_\mathrm{t}, \boldsymbol{u}\cdot\boldsymbol{n}_\mathrm{t})\ .$$

In addition to 3-D structural features, 3-D features derived from texture, for example, coregistered color images or scan reflectivities, are widely used for registration [31.56, 57]. Figure 31.15 shows scale invariant feature transform (SIFT) features extracted from a 3-D scanner with calibrated reflectivity values.

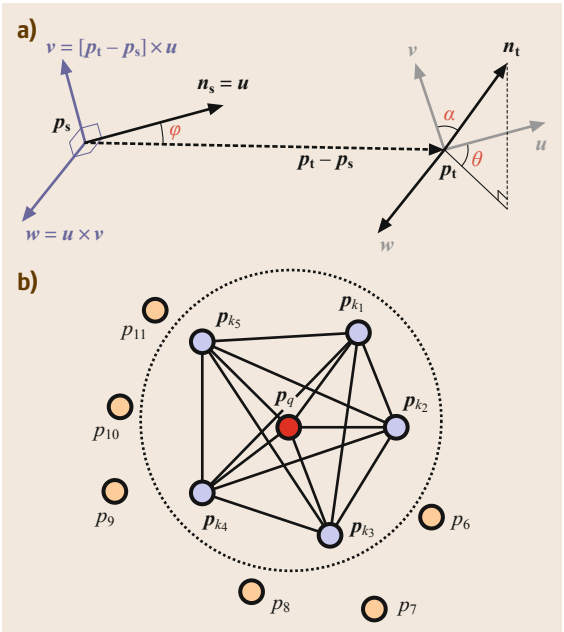


Fig.31.14a,b Angular features in PFHs (**a**) and the definition of the two regions (**b**)

Avidbots Ex1016
Page 831 of 2154



Fig. 31.15 Sift features extracted from the reflectivity values of a 3-D laser scan

Planes. A planar surface may only be described by the infinite surface as given by the equation, but it may also include a description of the boundary of the surface patch. Convenient representations for robotics applications are lists of the 3-D points $\{(x_i, y_i, z_i)\}$ that form the patch boundary, or polylines, which represent the boundary by a set of connected line segments. A polyline is represented by the sequence of 3-D points $\{(x_i, y_i, z_i)\}$ that form the vertices that join the line segments.

Plane extraction, or plane fitting, is the problem of modeling a given 3-D point cloud as a set of planes that ideally explain every data point. The RANSAC algorithm is one possible algorithm. When adapted for plane finding, this algorithm selects three 3-D points at random (although exploiting some locality to the point selection algorithm can improve the efficiency of the algorithm). These three points determine a plane with a parameter vector $\boldsymbol{a}$. Test all points $\{\boldsymbol{p}\}$ in the set for belonging to the plane ($|\,\boldsymbol{p}_i \cdot \boldsymbol{a}\,| < \tau$). If enough points are close to the plane, then potentially a plane has been found. These points should also be processed to find a connected set, from which a more accurate set of plane parameters can be estimated using the least-squares algorithm given above. If a planar patch is successfully found, the points that lie in that plane are removed from the dataset. The random selection of three points then continues until no more planes are found (a bound on how many tries to make can be estimated). For example, *Schnabel* et al. [31.58] have adapted RANSAC for plane extraction and found that the algorithm performs precise and fast plane extraction, but only if the parameters have been fine-tuned properly. For their optimization, they use information , which is not readily available in point cloud data, such as normals, neighboring relations, and outlier ratios.

A further standard method for plane detection is region growing, based on a seed patch. When the scene largely consists of planes, a particularly simple approach is based on selecting a previously unused point and the set of points $\{\boldsymbol{p}_i = (x_i, y_i, z_i)\}$ in its neighborhood. A plane is fit to these points using the least-squares method (cf. normal computation). This hypothesized plane then needs to be tested for reasonableness by 1) examining the smallest eigenvalue – it should be small and of the order of the square of the expected noise level and 2) ensuring that most of the 3-D points in the fitted set lie on the plane ($|\,\boldsymbol{p}_i \cdot \boldsymbol{a}\,| < \tau$).

Larger planar regions are *grown* by locating new adjacent points $\boldsymbol{p}_i$ that lie on the plane ($|\,\boldsymbol{p}_i \cdot \boldsymbol{a}\,| < \tau$). When enough of these are found, the parameters $\boldsymbol{a}$ of the plane are re-estimated. Points on the detected plane are removed and the process is repeated with a new seed patch. This process continues until no more points can be added. Complete descriptions of planar feature extraction with region growing are given in [31.59].

Bauer and *Polthier* use the radon transform to detect planes in volume data [31.60]. The idea and the speed of the algorithm are similar to that of the standard Hough transform. The Hough transform [31.61] is a method for detecting parametrized objects. For the Hough transform, planes are represented in the Hesse normal form, using normal vectors. A plane is thereby given by a point $\boldsymbol{p}$ on the plane, the normal vector $\boldsymbol{n}$ that is perpendicular to the plane and the distance ρ to the origin

$$\rho = \boldsymbol{p} \cdot \boldsymbol{n} = p_x n_x + p_y n_y + p_z n_z = \rho\,.$$

Considering the angles between the normal vector and the coordinate system, the coordinates of $\boldsymbol{n}$ are factorized to

$$p_x \cdot \cos\theta \cdot \sin\varphi + p_y \cdot \sin\varphi \cdot \sin\theta + p_z \cdot \cos\varphi = \rho\,, \tag{31.16}$$

with θ is the angle of the normal vector on the xy-plane and φ the angle between the xy-plane and the normal vector in the z direction. φ, θ, and ρ define the 3-dimensional Hough space (θ, φ, ρ) such that each point in the Hough space corresponds to *one plane* in $\mathbb{R}^3$. To find planes in a point set, one calculates the Hough transform for each point. Given a point $\boldsymbol{p}$ in Cartesian coordinates, one finds all planes the point lies on, that is, find all the θ, φ, and ρ that satisfy (31.16). Marking these points in the Hough space, that is, leads to a 3-D sinusoid curve as shown in Fig. 31.16. The intersections of two curves in the Hough space denote the planes that are rotated around the line built by the two points. Consequently, the intersection of three curves

Avidbots Ex1016
Page 832 of 2154

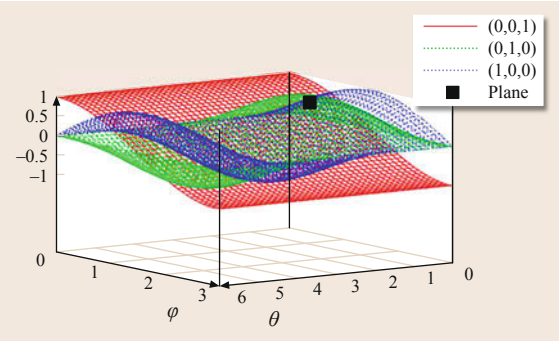


Fig. 31.16 Transformation of three points from $\mathbb{R}$ to the Hough space (θ, φ, ρ). The intersection of the curves (*marked in black*) depicts the plane spanned by the three points

in the Hough space corresponds to the polar coordinates defining the plane spanned by the three points. In Fig. 31.16, the intersection is marked in black. Given a set P of points in Cartesian coordinates, one transforms all points $\boldsymbol{p}_i \in P$ into the Hough space. The more curves intersect in $\boldsymbol{h}_j \in (\theta, \varphi, \rho)$, the more points lie on the plane represented by $\boldsymbol{h}_j$ and the higher is the probability that $\boldsymbol{h}_j$ is actually extracted from P.

The standard Hough transform is far too slow for plane detection in real-world data. A variant called randomized Hough transform is the method of choice when dealing with 3-D data due to its exceptional performance as far as runtime and quality are concerned [31.62].

Other plane extraction algorithms are highly specialized for a specific application and are not in widespread use for miscellaneous reasons. *Lakaemper* and *Latecki* [31.63] used an expectation maximization (EM) algorithm to fit planes that are initially randomly generated, *Wulf* et al. [31.64] detected planes relying on the specific properties of a sweeping laser scanner and *Yu* et al. [31.65] developed a clustering approach to solve the problem.

Triangulated Surfaces. Most commonly, surfaces are approximated by polygonal meshes, particularly triangle meshes, a standard data structure in computer graphics to represent 3-D objects. This representation describes an object or scene by a set of triangular patches. More general polygonal surface patches or even various smooth surface representations are also used, but triangles are most commonly used because they are simpler and there are inexpensive PC graphics cards that display triangles at high speed.

The triangles can be large (e.g., when representing planar surfaces) or small (e.g., when representing curved surfaces). The size chosen for the triangles reflects the accuracy desired for representing the object or scene surfaces. The triangulated surface might be complete in the sense that all observable scene or objects surfaces are represented by triangles, or there might be disconnected surface patches with or without internal holes. For grasping or navigation, you do not want any unrepresented scene surface to lie in front of the represented portion of the surface where a gripper or vehicle might collide with it. Hence, we assume that the triangulation algorithms produce patch sets that, if completely connected at the edges, implicitly bound all real scene surfaces. Figure 31.17 shows an example of a triangulated surface and the original point cloud.

The de-facto standard is the marching cubes method introduced by *Lorensen* nad *Cline* [31.66]. This algorithm subdivides the scanned volume into cubic cells or voxels. For each cell, the intersections between the cell edges and the surface are calculated. Precalculated surface patterns are then used to generate a local triangle mesh approximation. An example of such patterns are given in Fig. 31.18. To interpolate the intersections, implicit continuous surface representations like planes or splines are fitted to the local data using least-squares fits [31.67, 68]. A feature of the marching cubes algorithm is that it produces more triangles than are needed to represent an object. Hence, several mesh simplification algorithms have been introduced over the past years. Most of them define error metrics that indicate the error that a certain operation causes to the model, that is, the removal of an edge [31.69, 70]. To optimize the model, the edges causing the minimal error to the

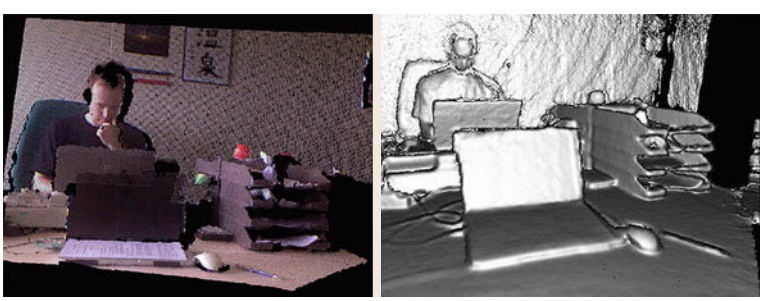


Fig. 31.17 **(a)** Image acquired with a kinect-like sensor. **(b)** Reconstruction with kinect fusion

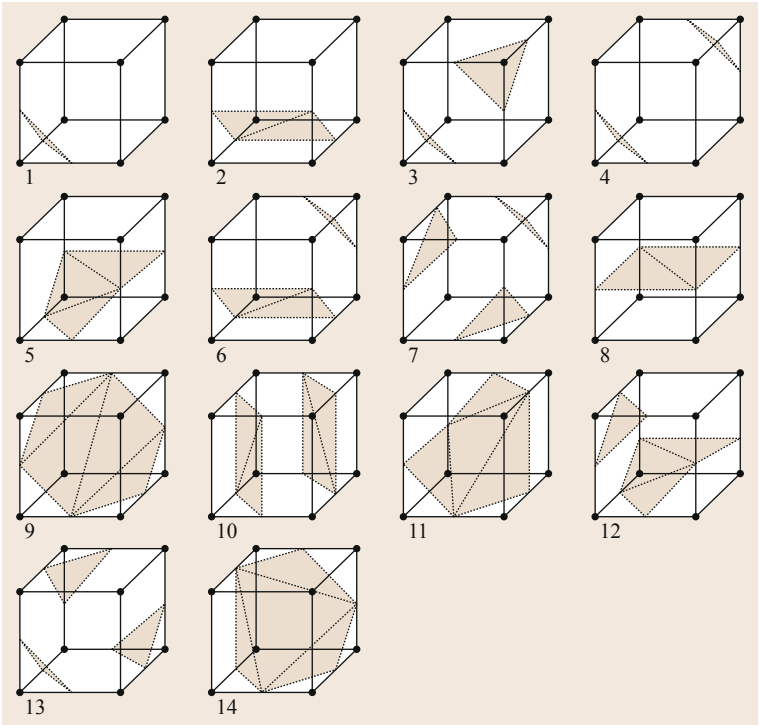


Fig. 31.18 Pattern for meshing 3-D voxels. 256 combinations are possible, but it is sufficient to model 16, iff symmetries are considered

topology are removed iteratively. Since after each edge removal new vertices have to be inserted into the mesh, the initial topology can be altered.

The kinect fusion approach modifies Hoppe's distance function [31.71]. It exploits the properties of the depth image (rectangular) to calculate normals and the associated planes. The method is massively parallelized by a GPU (graphics processing unit) implementation and can reconstruct and register meshes in real time.

3-D Lines. The 3-D lines where planar surfaces meet are features that can be easily detected by both stereo and range sensors. These features occur commonly in built environments (e.g., where walls, floors, ceilings, and doorways meet, around the edges of wall structures like notice boards, at the edges of office and warehouse furniture, etc.). They are also common on manmade objects. In the case of stereo, changes of surface shape or coloring are detected as edges, which can be matched in the stereo process to directly produce the 3-D edge. In the case of a range sensor, planar surfaces can be easily extracted from the range data (see the next section) and adjacent planar surfaces can be intersected to give the edges.

The most straightforward representation for 3-D lines is the set of points $\boldsymbol{x} = \boldsymbol{p} + \lambda \boldsymbol{v}$ for all λ, where $\boldsymbol{v}$ is a unit vector. This has 5 degrees of freedom; more complex representations for example, with 4 degrees of freedom exist [31.10].

Voxels. The voxel (volume pixel) approach represents the 3-D world by 3-D boxes/cells that indicate where there is a scene structure and where there is free space. The simplest representation is a 3-D binary array, encoded as 1 for having a structure and 0 for free space. This can be quite memory intensive, and also requires a lot of computation to check many voxels for content. A more complex but more compact representation is the hierarchical representation called the octree [31.72]. This divides the entire (bounded) rectangular space into eight rectangular subspaces called octants (Fig. 31.19). A tree data structure encodes the content of each octant as empty, full or mixed. Mixed octants are then subdivided into eight smaller rectangular octants, encoded as subtrees of the larger tree. Subdivision continues until some minimum octant size is reached. Determining whether a voxel is empty, full, or mixed depends on the sensor used, however, if no 3-D data points are located in the volume of a voxel, then it is likely to be empty. Similarly, if many 3-D points are present, then the voxel is likely to be full. Currently, many implementations using octrees for range data are available [31.73, 74]. Furthermore, these voxel representations are the basis of surface/mesh reconstruction algorithms as described earlier.

Avidbots Ex1016
Page 834 of 2154

For the purpose of robot navigation, localization or grasping, only the surface and free-space voxels need to be marked accurately. The interior of objects and scene structure are largely irrelevant.

Straight Lines. While straight lines are common in man-made scenes, direct extraction from 3-D datasets is not easy. The main source of the difficulty is that the 3-D sensors often do not acquire good responses at edges of surfaces. For this reason, most 3-D line detection algorithms are indirect, whereby planes are first detected, for example, using the method of the previous section, and then adjacent planes are intersected. Adjacency can be tested by finding paths of connected pixels that lead from one plane to the other. If planes 1 and 2 contains points $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$ and have surface normals $\boldsymbol{n}_1$ and $\boldsymbol{n}_2$, respectively, then the resulting intersection line has equation $\boldsymbol{x} = \boldsymbol{a} + \lambda \boldsymbol{d}$ where $\boldsymbol{a}$ is a point on the line and $\boldsymbol{d} = \frac{\boldsymbol{n}_1 \times \boldsymbol{n}_2}{||\boldsymbol{n}_1 \times \boldsymbol{n}_2||}$ is the line direction.

There are an infinite number of possible points $\boldsymbol{a}$, which can be found by solving the equations $\boldsymbol{a}'\boldsymbol{n}_1 = \boldsymbol{p}_1'\boldsymbol{n}_1$ and $\boldsymbol{a}'\boldsymbol{n}_2 = \boldsymbol{p}_2'\boldsymbol{n}_2$. A reasonable third constraint that obtains a point near $\boldsymbol{p}_2$ is the equation $\boldsymbol{a}'\boldsymbol{d} = \boldsymbol{p}_2'\boldsymbol{d}$. This gives us an infinite line. Most practical applications require a finite segment. The endpoints can be estimated by (1) finding the points on the line that lie close to observed points in both planes and then (2) finding the two extremes of those points. On the other hand, finding straight 3-D lines can be easier with a stereo sensor, as these result from matching two straight 2-D image lines.

31.3.3 Multiple-View Registration

A globally consistent representation of a robot's environment is crucial for many robotic applications. Equipped with a 3-D depth-perceiving sensor, many mobile systems gather spatial information about their local 3-D environments. Any iterative application of matching algorithms leads to inconsistencies due to sensing errors and due to the inaccuracies in the matching procedures itself. To avoid these problems, global matching algorithms are needed, taking global correspondences between range sensor data into account. Simultaneous localization and mapping (SLAM) algorithms as discussed in Chap. 46 solve a very similar problem. In addition to registering multiple views, they estimate a map. Multiple-view registration also relates to bundle adjustment in the photogrammetry community and structure from motion (SFM).

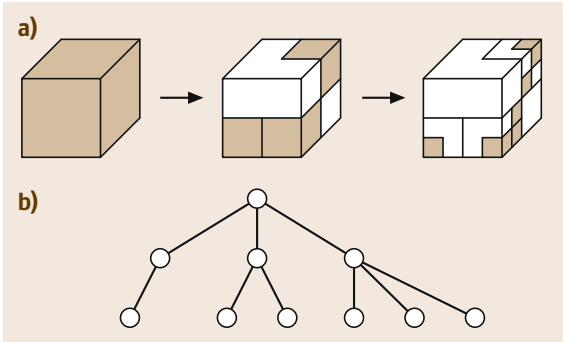


Fig. 31.19 (a) Spatial subdivisions of an octree up to level 3. Occupied leaf nodes are shaded grey. (b) The corresponding tree structure of the sparse data structure

If n-views have to be registered, any sequential application of a two point-set registration method will accumulate errors, and therefore the registration algorithm (Sect. 31.3.1) has to be extended. The global error function becomes

$$E = \sum_{l \to k} \sum_{i} \left\| (\mathbf{R}_l \boldsymbol{m}_{l,i} + \boldsymbol{t}_l) - (\mathbf{R}_k \boldsymbol{d}_{k,i} + \boldsymbol{t}_k) \right\|^2 , \tag{31.17}$$

where all views have their unique pose $(\mathbf{R}, \boldsymbol{t})$. After the point pairs for all overlapping views (l, k) have been found, (31.17) is minimized. Unfortunately, a closed-form solution for minimizing (31.17) is not know, but a small angle approximation or the helix transform yield a system of linear equations that can be solved by Choleskey decomposition [31.49]. In an ICP-like fashion after every transformation new point pairs have to be found (Algorithm 31.2).

Algorithm 31.2 The globally consistent ICP algorithm

1: **for** $i = 0$ to *maxIterations* **do**
2: find the closest point within a range $d_{\max}$ of every pair of overlapping 3-D point clouds (l, k).
3: Calculate n transformations $(\mathbf{R}, \boldsymbol{t})$ simultaneously that minimize the error function (31.17)
4: Apply the n transformations found in step 4 to all data sets.
5: Compute the difference of the quadratic error, that is, compute the difference of the value $\|E_{i-1} - E_i\|$ before and after the application of the transformation. If this difference falls below a threshold ε, terminate.
6: **end for**

A probabilistic SLAM-like notation of (31.17) was formulated in [31.75] for 2-D range scans. For each pose $\boldsymbol{X}$, the term $\bar{\boldsymbol{X}}$ denotes a pose estimate, and $\Delta\boldsymbol{X}$ is the pose error. The positional error of two poses $\boldsymbol{X}_j$

Avidbots Ex1016
Page 835 of 2154

and X_k is described by

$$E_{j,k} = \sum_{i=1}^{m} \left\| X_j \oplus d_i - X_k \oplus m_i \right\|^2 .$$

Here, $\oplus$ is the compounding operation that transforms a point into the global coordinate system. For small pose differences, $E_{j,k}$ can be linearized by using a Taylor expansion. With the linearized error metric $E'_{j,k}$ and the Gaussian distribution $(\bar{E}_{j,k}, \mathbf{C}_{j,k})$ a Mahalanobis distance that describes the global error of all the poses is constructed

$$\begin{aligned} \mathbf{W} &= \sum_{j \to k} \left(\bar{E}_{j,k} - E'_{j,k}\right)^{\mathrm{T}} \mathbf{C}_{j,k}^{-1} \left(\bar{E}'_{j,k} - E'_{j,k}\right) \\ &= \sum_{j \to k} \left[\bar{E}_{j,k} - \left(X'_j - X'_k\right)\right] \mathbf{C}_{j,k}^{-1} \left[\bar{E}'_{j,k} - \left(X'_j - X'_k\right)\right], \end{aligned} \tag{31.18}$$

which can be solved efficiently using iterative least squares techniques, for example, Levenberg–Marquadt or conjugate gradient methods [31.76, 77]. The covariances are computed from point pairs. In the presence of correct covariances, (31.18) is minimized once. In case of scan matching, new pose estimates yield new closest point pairs and in turn new covariances. Iterating the process of calculating point pairs and minimization yields a stable algorithm that converges rapidly. The probabilisty notation and the global ICP notation are very similar [31.49]. The solution for 2-D range scans has been extended to to 6-DOF in [31.78].

31.3.4 Model Matching

Model matching is the process of matching some stored representation to some observed data. In the case discussed here, we assume that both are 3-D representations. Furthermore, we assume that the representations being matched are both of the same type, for example, 3-D model and scene lines. (While different types of data can also be matched, we ignore these more specialized algorithms here.)

A special case of matching is when the two structures being matched are both scene or model surfaces. The algorithm used for matching depends on the complexity the structures being matched. If the structures being matched are extended geometric entities such as planes or 3-D lines, then a discrete matching algorithm like the Interpretation Tree algorithm [31.79] can be used. It is suitable for matching small numbers (e.g., less than about 20–30) discrete objects, such as vertical edges seen in 2-D or 3-D. If there are M model and D data objects, then potentially there are M^D different matches. The key to efficient matching is to identify pairwise constraints that eliminate unsuitable matches. Constraints between pairs of model features and pairs of data features also greatly reduce the matching space. If the constraints eliminate enough features, a polynomial time algorithm results. The core of the algorithm is defined as follows. Let $\{m_i\}$ and $\{d_j\}$ be the sets po model and data features to be matched, $u(m_i, d_j)$ is true if m_i and d_j are compatible features, $b(m_i, m_j, d_k, d_l)$ is true if the four model and data features are compatible and T is the minimum number of matched features before a successful match is declared. `Pairs` is the set of successfully matched features. The function `truesizeof` counts the number of actual matches in the set, disregarding matches with the wildcard * which matches anything.

Algorithm 31.3

```
pairs=it(0,{})
if truesizeof(pairs) >= T, then success

function pairs=it(level,inpairs)
  if level >= T, then return inpairs
  if M-level+truesizeof(inpairs) < T
    then return {} % can never succeed
  for each d_i % loopD start
    if not u(m_level,d_i), then
                              continue loopD
    for each (m_k,d_l) in inpairs
      if not b(m_level,m_k,d_i,d_l)
        then continue loopD
    endfor
    % have found a~successful new pair
                                    % to add
    pairs = it(level+1,
                union(inpairs,(m_level,d_i)))
    if truesizeof(pairs) >= T, then return
  endfor % loopD end

  % no success, so try wildcard
  it(level+1,union(inpairs,(m_level,*)))
```

31.3.5 Relative Pose Estimation

Central to many tasks is the estimation of the coordinate system relative position or pose transformation between two coordinate systems. For example, this might be the pose of a scanner mounted on a mobile vehicle relative to scene landmarks. Or, it might be the relative pose of some scene features as observed in two views taken from different positions.

We present here three algorithms that cover most instances of the pose estimation process, which differ slightly based on the type of feature being matched.

Avidbots Ex1016
Page 836 of 2154

Point Set Relative Pose Estimation

The ICP algorithm can be used relative pose estimation as well (Sect. 31.3.1).

Straight Line Relative Pose Estimation

If 3-D lines are the features that are extracted, then the relative pose transformation can be estimated as follows. Assume N paired lines. The first set of lines is described by direction vectors $\{\boldsymbol{e}_i\}$ and a point on each line $\{\boldsymbol{a}_i\}$. The second set of lines is described by direction vectors $\{\boldsymbol{f}_i\}$ and a point on each line $\{\boldsymbol{b}_i\}$. In this algorithm, we assume that the direction vectors on the matched segments always point the same direction (i. e., are not inverted). This can be achieved by exploiting some scene constraints, or trying all combinations and eliminating inconsistent solutions. The points $\boldsymbol{a}_i$ and $\boldsymbol{b}_i$ need not correspond to the same point after alignment. The desired rotation matrix $\mathbf{R}$ minimizes $\sum_i || \mathbf{R}\boldsymbol{e}_i - \boldsymbol{f}_i ||^2$. Construct the $3 \times N$ matrices $\mathbf{E}$ that consists of the vectors $\{\boldsymbol{e}_i\}$ stacked up. Construct the $3 \times N$ matrices $\mathbf{F}$ in a similar way from the vectors $\{\boldsymbol{f}_i\}$. Compute the singular value decomposition $\text{svd}(\mathbf{F}\mathbf{E}') = \mathbf{U}'\mathbf{D}\mathbf{V}'$. Compute the rotation matrix $\mathbf{R} = \mathbf{V}\mathbf{U}'$. The translation estimate $\boldsymbol{t}$ minimizes the sum of the square of the distances λ_i between the rotated points $\boldsymbol{a}_i$ and corresponding line $(\boldsymbol{f}_i, \boldsymbol{b}_i)$. Define matrix $\mathbf{L} = \sum_i (\mathbf{I} - \boldsymbol{f}_i \boldsymbol{f}_i')'(\mathbf{I} - \boldsymbol{f}_i')$. Define the vector $\boldsymbol{n} = \sum_i (\mathbf{I} - \boldsymbol{f}_i \boldsymbol{f}_i')'(\mathbf{I} - \boldsymbol{f}_i \boldsymbol{f}_i')(\mathbf{R}\boldsymbol{a}_i - \boldsymbol{b}_i)$. Then the translation is $\boldsymbol{t} = -\mathbf{L}^{-1}\boldsymbol{n}$.

Plane Relative Pose Estimation

Finally, if planes are the 3-D features extracted for matching, then the relative pose transformation can be estimated as follows. Assume N paired planes. The first set of planes is described by surface normals $\{\boldsymbol{e}_i\}$ and a point on each plane $\{\boldsymbol{a}_i\}$. The second set of planes is described by surface normals $\{\boldsymbol{f}_i\}$ and a point on each plane $\{\boldsymbol{b}_i\}$. Here we assume that the surface normals always point outward from the surface. The points $\boldsymbol{a}_i$ and $\boldsymbol{b}_i$ need not correspond to the same point after alignment. The desired rotation matrix $\mathbf{R}$ minimizes $\sum_i || \mathbf{R}\boldsymbol{e}_i - \boldsymbol{f}_i ||^2$. Construct the $3 \times N$ matrices $\mathbf{E}$ that consists of the vectors $\{\boldsymbol{e}_i\}$ stacked up. Construct the $3 \times N$ matrices $\mathbf{F}$ in a similar way from the vectors $\{\boldsymbol{f}_i\}$. Compute the singular value decomposition $\text{svd}(\mathbf{F}\mathbf{E}') = \mathbf{U}'\mathbf{D}\mathbf{V}'$ [31.45]. Compute the rotation matrix $\mathbf{R} = \mathbf{V}\mathbf{U}'$. The translation estimate $\boldsymbol{t}$ minimizes the sum of the square of the distances λ_i between the rotated point $\boldsymbol{a}_i$ and the corresponding plane $(\boldsymbol{f}_i, \boldsymbol{b}_i)$. Define matrix $\mathbf{L} = \sum_i \boldsymbol{f}_i \boldsymbol{f}_i'$. Define the vector $\boldsymbol{n} = \sum_i \boldsymbol{f}_i \boldsymbol{f}_i'(\mathbf{R}\boldsymbol{a}_i - \boldsymbol{b}_i)$. Then the translation is $\boldsymbol{t} = -\mathbf{L}^{-1}\boldsymbol{n}$.

In all of the calculations described above, we assumed normally distributed errors. For techniques to robustify these sorts of calculations, see *Zhang* [31.80].

31.3.6 3-D Applications

This section links the techniques presented above to the robotics applications of 3-D localization of parts for robot manipulation, self-localization of robot vehicles and scene understanding for robot navigation. The robotics tasks mentioned here are discussed in more detail in other chapters in the series. While this chapter focusses on robotics applications, there are many other 3-D sensing applications. An area of much current research is that of acquiring 3-D models, particularly for reverse engineering of mechanical parts [31.81], historical artifacts [31.82], buildings [31.83] and people for computer games and movies (Cyberware Whole Body X 3-D Scanner).

The key tasks in robot manipulation are:

1. Identification of grasping points (Chaps. 37 and 38),
2. Identification of a collision free grasp (Chaps. 37 and 38),
3. Recognition of parts to be manipulated (Chap. 32) and
4. Position estimation of parts for manipulation (Chaps. 32 and 42).

The key tasks in robot navigation and self-localization are:

4. Identification of a navigable groundplane (Sect. 31.4),
5. Identification of a collision free path (Chap. 47),
6. Identification of landmarks (Chap. 45) and
7. Estimation of vehicle location (Chap. 53).

The mobile and assembly robotics tasks link together rather naturally. Tasks 1 and 5 have a connection, when we consider these tasks in the context of unknown parts or paths. Part grasping requires finding regions on a part that are graspable, which usually means locally planar patches that are large enough that a gripper can make good contact with them. Similarly, navigation usually requires smooth ground regions that are large enough for the vehicle – again locally planar patches. Both tasks are commonly based on triangulated scene methods to represent the data, from which connected regions of nearly coplanar patches can be extracted. The main difference between these two tasks is the groundplane detection task is looking for a larger patch, that must be on the *ground* and upward facing.

Tasks 2 and 6 require a method of representing empty space along the proposed trajectory of the gripper contacts or the vehicle. The voxel representation is good for this task.

Avidbots Ex1016
Page 837 of 2154

Tasks 3 and 7 are model matching tasks and can use the methods of Sect. 31.3.3 to match observed scene features to prestored models of known parts or scene locations. Commonly used features are large planar surfaces, 3-D edges and 3-D feature points.

Tasks 4 and 8 are pose estimation tasks and can use the methods of Sect. 31.3.5 to estimate the pose of the object relative to the sensor or vehicle (i. e., sensor) relative to the scene. Again, commonly used features are large planar surfaces, 3-D edges and 3-D feature points.

31.4 Navigation and Terrain Classification and Mapping

One of the more compelling uses for range data is for navigation of mobile robot vehicles. Range data provides information about obstacles and free space for the vehicle, in a direct geometric form. Because of the real-time constraints of navigation, it is often impractical to reconstruct a full 3-D model of the terrain using the techniques presented in this chapter. Instead, most systems use an *elevation model*. An elevation model is a tesselated 2-D representation of space, where at each cell there is information about the distribution of 3-D points in the cell. In its simplest incarnation, the elevation map just contains the mean height of range points above the nominal ground plane (Fig. 31.20). This representation is sufficient for some indoor and urban environments; more sophisticated versions that determine a local plane, scatter of points in the cell, etc., are useful for more complicated off-road driving. Elevation maps marked with obstacles have obvious utility for planning a collision-free path for the vehicle.

31.4.1 Mobile Mapping

Laser range scanning provides an efficient way to actively acquire accurate and dense 3-D point clouds of object surfaces or environments. Mobile scanning is currently used for modeling in architecture and agriculture as well as urban and regional planning. Modern systems like the Riegl VMX-450 and the Lynx mobile mapper as produced by Optech work along the same basic principal. They combine a highly accurate global positioning system (GPS), a high precision inertial measurement unit (IMU) and the odometry of the vehicle to compute the fully timestamped trajectory. Using a process called motion compensation this trajectory is then used to *unwind* the laser range measurements that were acquired by the 2-D laser scanner also mounted on the vehicle. The quality of the resulting point cloud depends on several factors:

- The calibration of the entire system, that is, the accuracy to which the position and orientation of each individual sensor in relation to the vehicle has been determined.
- The accuracy of the external positioning sensors, that is, the GPS, IMU and odometry.
- The availability of the GPS, as it may suffer temporary blackouts under bridges, in tunnels and between high-rises.
- The accuracy of the laser scanner itself.

The movement of the mobile laser scanner between time t_0 and t_n creates a trajectory $T = \{\boldsymbol{V}_0, \ldots, \boldsymbol{V}_n\}$, where $\boldsymbol{V}_i = (t_{x,i}, t_{y,i}, t_{z,i}, \theta_{x,i}, \theta_{y,i}, \theta_{z,i})$ is the 6-DOF (degree of freedom) pose of the vehicle at time t_i with $t_0 \leq t_i \leq t_n$. Using the trajectory of the vehicle a 3-D representation of the environment can be obtained by *unwinding* the laser measurements M to create the final map P. However, sensor errors in odometry, IMU and GPS as well as systematic calibration errors and the accumulation of pose errors during temporary GPS outages degrade the accuracy of the trajectory and therefore the point cloud quality. Furthermore, please note that modern systems easily create thouthands of 3-D scan slices.

These mobile mapping systems extend the early work on 3-D reconstruction with two 2-D lasers under exploitation of the robot motion [31.84]. One laser scanner is scanning horizontal, one is mounted vertically. The resultant point cloud is typically registered using the pose of the robot as corrected by the 2-D SLAM algorithm, rather than any of the 3-D registration techniques covered in this chapter. The current state of the art developed by *Bosse* et al. [31.85] for improving overall map quality of mobile mappers in the robotics

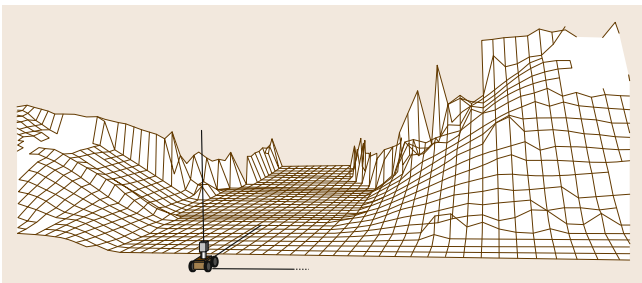


Fig. 31.20 Elevation map in urban terrain. Each cell holds the height of the terrain at that point. More extensive features can also be incorporated: slope, point variance, etc.

Avidbots Ex1016
Page 838 of 2154

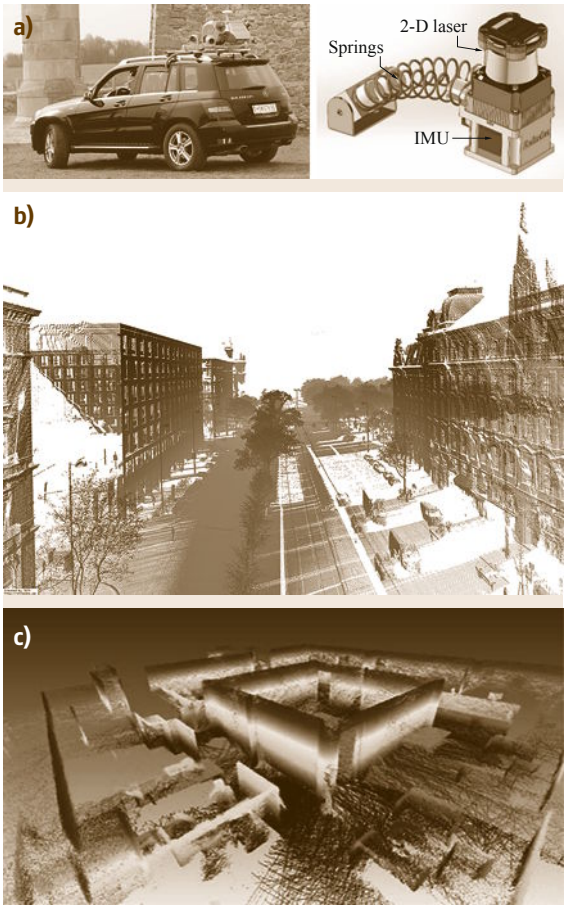


Fig. 31.21 **(a)** Mobile mapping car and **(b)** Zedebee handheld mapper. **(c)** 3-D point cloud acquired with the Riegl VMX-250 system. **(d)** 3-D point cloud from Zedebee

community is to coarsely discretize the time. This results in a partition of the trajectory into subscans that are treated rigidly. Then rigid registration algorithms like the ICP and other solutions to the SLAM problem are employed.

Figure 31.21 shows a car equipped with the Riegl VMX-450 mobile mapping system of the company Riegl laser measurement systems and the Zebedee handhald mapper from CSIRO. Figure 31.21 shows typical resulting 3-D point clouds as well.

31.4.2 Urban Navigation

In urban navigation, the environment is structured, with roads, buildings, sidewalks, and also moving objects – people and other vehicles. There are two main challenges: how to register laser scans from a fast-moving vehicle for consistent mapping, and how to detect moving objects using range scans (of course, other methods

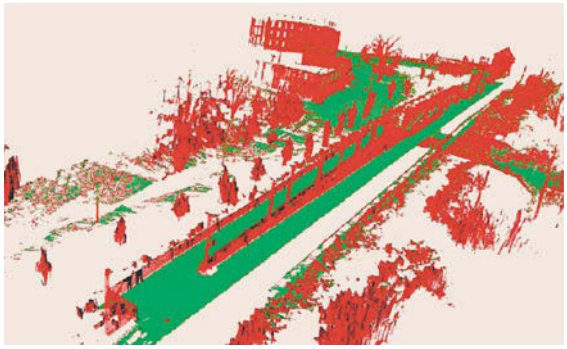


Fig. 31.22 Elevation map of an urban scene, using 10×10 cm cells. Obstacles in *red*, ground plane in *green* (after [31.88])

are also used for detecting moving objects, for example, appearance-based vision).

Outdoor vehicles can use precision GPS, inertial measurement units, and wheel odometry to keep track of their position and orientation, typically with an extended Kalman filter. This method is good enough to obviate the need for precise registration matching among scans, as long as the motion model of the vehicle, and timing from the range scanner, is used to place each scan reading in its proper position in the world model. This method also works in relatively easy offroad terrain such as in the DARPA (Defense Advanced Research Projects Agency) Grand Challenge [31.86]. In all cases, the reduction of pose estimation error is critical for good performance [31.87].

Once scan readings are registered using the vehicle pose estimation, they can be put into an elevation map, and obstacles detected using the slope and vertical extent of the range readings in the cells of the map. A complication is that there may be multiple levels of elevation in an urban setting, for example, an overpass would not be an obstacle if it were high enough. One proposal is to use multiple elevation clusters within each cell; this technique is called a *multilevel surface map* (MLS, [31.88]). Each cell in the map stores a set of surfaces represented by a mean height and variance. Figure 31.22 shows an MLS with a cell size of 10 cm^2, with ground plane and obstacles marked.

For dynamic objects, real-time stereo at 15–30 Hz can capture the motion of the objects. When the stereo rig is fixed, range background subtraction isolates just the moving objects [31.89]. When the rig is on a moving vehicle, the problem is more difficult, since the whole scene is moving with respect to the rig. It can be solved by estimating the motion of the rig with respect to the dominant rigid background of the scene. Let R, t be the motion of the rig between two frames, estimated by extracting features and matching them across the two



Fig.31.23a–c Independent motion detection from a moving platform. Reference image on panel **(a)** is forward-projected using the motion homography to the image on panel **(b)**; image on panel **(c)** is the difference with the actual image

temporal frames, using the techniques of Chap. 46. The homography $H(R,t)$ of (31.11) provides a direct projection of the disparity vectors $p_0 = [x_0, y_0, d_0, 1]$ of the first frame to their correspondences $H(R,t)p_0$ under R,t in the second frame. Using the homography allows the points in the reference frame to be directly projected onto the next frame, without translating to 3-D points. Figure 31.23 shows the projected pixels under rigid motion from a reference scene. The difference between the projected and actual pixels gives the independently moving objects (from [31.90]).

31.4.3 Rough Terrain

Rough outdoor terrain presents two challenges:

- There may be no extensive ground plane to characterize driveability and obstacles.
- Vegetation that is pliable and driveable may appear as an obstacle in range images.

Figure 31.24 shows a typical outdoor scene, with a small (1 m) robot driving through vegetation and rough ground [31.91]. Range data from stereo vision on the robot will see the top of the vegetation and some ground points below. The elevation model can be extended to look at *point statistics* within each cell, to capture the notion of a local ground plane and penetrability related to vegetation. In [31.92], for example, the set of proposed features includes:

- Major plane slope using a robust fit (Sect. 31.3.2, *3-D Feature Representations and Extraction*)
- Height difference of max and min heights
- Points above the major plane
- Density: ratio of points in the cell to rays that pass through the cell.

The density feature is interesting (and expensive to compute), and attempts to characterize vegetation such a grass or bushes, by looking at whether range readings penetrate an elevation cell. The idea of using vegetation permeability to range readings has been discussed in several other projects on off-road driving [31.94–96].

Elevation map cells can be characterized as obstacles or driveable through learning or hand-built classifiers. Among the learning techniques are neural nets [31.92] and Gaussian mixture models with



Fig. 31.24 Rough terrain, no ground plane, driveable vegatation

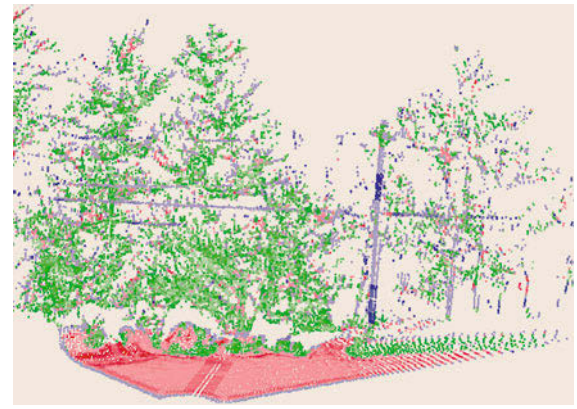


Fig. 31.25 Classification using point statistics. *Red* is planar surface, *blue* is thin linear surface, *green* is scattered penetrable surface (after [31.93])

Avidbots Ex1016
Page 840 of 2154

expectation-maximization learning [31.93]. The latter work also includes a lower level of interpretation, classifying surfaces into planar patches (ground plane, solid obstacles), linear features (telephone wires), and scattered features (vegetation). Figure 31.25 shows some results from a laser-scanned outdoor scene. Linear features such as telephone wires and the telephone pole are accurately determined, as well as vegetation with high penetrability.

Some additional problems occur in rough-terrain navigation. For planar laser rangefinders that are swept over the terrain by vehicle motion, the precision of vehicle pose estimation is important for accurate reconstruction. Attitude errors of less than 0.5° can cause false positives in obstacle detection, especially for sweeps far ahead of the vehicle. In [31.87], this problem is solved by looking at the time of each laser reading, and noting a correlation between height errors and time difference in the readings.

Negative obstacles (ditches and cliffs) are difficult to detect with range information, because the sensor may not see the bottom of the obstacle. This is especially true for vehicle-mounted sensors that are not very high off the ground, and that are looking far ahead. Negative obstacles can be inferred when there is a gap in the ground plane, and a plane slanted upward at the back edge of the gap. Such artifacts can be efficiently found using column search on the disparity image [31.97].

31.5 Conclusions and Further Reading

Range sensing is an active and expanding field of research in robotics. The presence of new types of devices – flash ladars, multibeam ladars, on-camera stereo processing – and the continuing development of robust algorithms for object reconstruction, localization and mapping has helped to bring applications out of the laboratory and into the real world. Indoor navigation with ladars is already being exploited in commercial products (for example, [31.98]). As the basic capabilities become more robust, researchers are looking to perform useful tasks, such as fetching items or doing dishes [31.99].

Another set of challenges are found in less benign environments, such as urban and off-road driving (DARPA Grand Challenge and Urban Challenge [31.86]). Stereo vision and laser rangefinding also will play a role in helping to provide autonomy for a new generation of more-capable robotic platforms that rely on walking for locomotion [31.100]. The challenges are dealing with motion that is less smooth than wheeled platforms, environments that contain dynamic obstacles, and task-oriented recognition of objects.

References

31.1 H. Houshiar, J. Elseberg, D. Borrmann, A. Nüchter: A study of projections for key point based registration of panoramic terrestrial 3D laser scans, J. Geo-Spat. Inf. Sci. **18**(1), 11–31 (2015)

31.2 Velodyne: High definition lidar, http://velodynelidar.com/ (2015)

31.3 R. Stettner, H. Bailey, S. Silverman: Three-dimensional flash Ladar focal planes and time-dependent imaging, Int. J. High Speed Electron. Syst. **18**(2), 401–406 (2008)

31.4 S.B. Gokturk, H. Yalcin, C. Bamji: A time-of-flight depth sensor – system description, issues and solutions, Computer Vis. Pattern Recognit. Workshop (CVPRW) (2004)

31.5 T. Oggier, M. Lehmann, R. Kaufmannn, M. Schweizer, M. Richter, P. Metzler, G. Lang, F. Lustenberger, N. Blanc: An all-solid-state optical range camera for 3D-real-time imaging with sub-centimeter depth-resolution (SwissRanger), Proc. SPIE **5249**, 534–545 (2003)

31.6 U. Wong, A. Morris, C. Lea, J. Lee, C. Whittaker, B. Garney, R. Whittaker: Red: Comparative evaluation of range sensing technologies for underground void modeling, Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS) (2011) pp. 3816–3823

31.7 D.D. Lichti: A review of geometric models and self-calibration methods for terrestrial laser scanner, Bol. Cienc. Géod. **16**(1), 3–19 (2010)

31.8 G. Iddan, G. Yahav: 3D imaging in the studio (and elsewhere...), Proc. SPIE 4298 (2003) pp. 48–55

31.9 TriDiCam GmbH: http://www.tridicam.de/en.html (2015)

31.10 R. Hartley, A. Zisserman: *Multiple View Geometry in Computer Vision* (Cambridge Univ. Press, Cambridge 2000)

31.11 S. Barnard, M. Fischler: Computational stereo, ACM Comput. Surv. **14**(4), 553–572 (1982)

31.12 D. Scharstein, R. Szeliski, R. Zabih: A taxonomy and evaluation of dense two-frame stereo correspondence algorithms, Int. J. Computer Vis. **47**(1–3), 7–42 (2002)

31.13 D. Scharstein, R. Szeliski: Middlebury College Stereo Vision Research Page, http://vision.middlebury.edu/stereo (2007)

31.14 R. Zabih, J. Woodfill: Non-parametric local transforms for computing visual correspondence, Proc. Eur. Conf. Comput. Vis., Vol. 2 (1994) pp. 151–158

31.15 O. Faugeras, B. Hotz, H. Mathieu, T. Viéville, Z. Zhang, P. Fua, E. Théron, L. Moll, G. Berry, J. Vuillemin, P. Bertin, C. Proy: Real time correlation based stereo: algorithm implementations and applications, Int. J. Computer Vis. **47**(1–3), 229–246 (2002)

31.16 M. Okutomi, T. Kanade: A multiple-baseline stereo, IEEE Trans. Pattern Anal. Mach. Intell. **15**(4), 353–363 (1993)

31.17 L. Matthies: Stereo vision for planetary rovers: stochastic modeling to near realtime implementation, Int. J. Comput. Vis **8**(1), 71–91 (1993)

31.18 R. Bolles, J. Woodfill: Spatiotemporal consistency checking of passive range data, Proc. Int. Symp. Robotics Res. (1993)

31.19 P. Fua: A parallel stereo algorithm that produces dense depth maps and preserves image features, Mach. Vis. Appl. **6**(1), 35–49 (1993)

31.20 H. Moravec: Visual mapping by a robot rover, Proc. Int. Jt. Conf. Artif. Intell. (IJCAI) (1979) pp. 598–600

31.21 A. Adan, F. Molina, L. Morena: Disordered patterns projection for 3D motion recovering, Proc. Int. Conf. 3D Data Process. Vis. Transm. (2004) pp. 262–269

31.22 Videre Design LLC: http://www.videredesign.com (2007)

31.23 Point Grey Research Inc.: http://www.ptgrey.com (2015)

31.24 C. Zach, A. Klaus, M. Hadwiger, K. Karner: Accurate dense stereo reconstruction using graphics hardware, Proc. EUROGRAPHICS (2003) pp. 227–234

31.25 R. Yang, M. Pollefeys: Multi-resolution real-time stereo on commodity graphics hardware, Int. Conf. Comput. Vis Pattern Recognit., Vol. 1 (2003) pp. 211–217

31.26 K. Konolige: Small vision system. Hardware and implementation, Proc. Int. Symp. Robotics Res. (1997) pp. 111–116

31.27 Focus Robotics Inc.: http://www.focusrobotics.com (2015)

31.28 TYZX Inc.: http://www.tyzx.com (2015)

31.29 S.K. Nayar, Y. Nakagawa: Shape from Focus, IEEE Trans. Pattern Anal. Mach. Intell. **16**(8), 824–831 (1994)

31.30 M. Pollefeys, R. Koch, L. Van Gool: Self-calibration and metric reconstruction inspite of varying and unknown intrinsic camera parameters, Int. J. Computer Vis. **32**(1), 7–25 (1999)

31.31 A. Hertzmann, S.M. Seitz: Example-based photometric stereo: Shape reconstruction with general, Varying BRDFs, IEEE Trans. Pattern Anal. Mach. Intell. **27**(8), 1254–1264 (2005)

31.32 A. Lobay, D.A. Forsyth: Shape from texture without boundaries, Int. J. Comput. Vis. **67**(1), 71–91 (2006)

31.33 Wikipedia: http://en.wikipedia.org/wiki/List_of_fastest-selling_products (2015)

31.34 K. Khoshelham, S.O. Elberink: Accuracy and resolution of kinect depth data for indoor mapping applications, Sensors **12**(5), 1437–1454 (2012)

31.35 P.J. Besl, N.D. McKay: A method for registration of 3D shapes, IEEE Trans. Pattern Anal. Mach. Intell. **14**(2), 239–256 (1992)

31.36 Y. Chen, G. Medioni: Object modeling by registration of multiple range images, Image Vis. Comput. **10**(3), 145–155 (1992)

31.37 Z. Zhang: *Iterative Point Matching for Registration of Free-Form Curves, Tech. Rep. Ser., Vol. RR-1658* (INRIA–Sophia Antipolis, Valbonne Cedex 1992)

31.38 S. Rusinkiewicz, M. Levoy: Efficient variants of the ICP algorithm, Proc. 3rd Int. Conf. 3D Digital Imaging Model. (2001) pp. 145–152

31.39 J.L. Bentley: Multidimensional binary search trees used for associative searching, Commun. ACM **18**(9), 509–517 (1975)

31.40 J.H. Friedman, J.L. Bentley, R.A. Finkel: An algorithm for finding best matches in logarithmic expected time, ACM Trans. on Math. Software **3**(3), 209–226 (1977)

31.41 M. Greenspan, M. Yurick: Approximate K-D tree search for efficient ICP, Proc. 4th IEEE Int. Conf. Recent Adv. 3D Digital Imaging Model. (2003) pp. 442–448

31.42 L. Hyafil, R.L. Rivest: Constructing optimal binary decision trees is NP-complete, Inf. Proc. Lett. **5**, 15–17 (1976)

31.43 N.J. Mitra, N. Gelfand, H. Pottmann, L. Guibas: Registration of point cloud data from a geometric optimization perspective, Proc. Eurographics/ACM SIGGRAPH Symp. Geom. Process. (2004) pp. 22–31

31.44 A. Nüchter, K. Lingemann, J. Hertzberg: Cached *k*-d tree search for ICP Algorithms, Proc. 6th IEEE Int. Conf. Recent Adv. 3D Digital Imaging Model. (2007) pp. 419–426

31.45 K.S. Arun, T.S. Huang, S.D. Blostein: Least-squares fitting of two 3-D point sets, IEEE Trans. Pattern Anal. Mach. Intell. **9**(5), 698–700 (1987)

31.46 B.K.P. Horn, H.M. Hilden, S. Negahdaripour: Closed-form solution of absolute orientation using orthonormal matrices, J. Opt. Soc. Am. A **5**(7), 1127–1135 (1988)

31.47 B.K.P. Horn: Closed-form solution of absolute orientation using unit quaternions, J. Opt. Soc. Am. A **4**(4), 629–642 (1987)

31.48 M.W. Walker, L. Shao, R.A. Volz: Estimating 3-d location parameters using dual number quaternions, J. Comput. Vis. Image Underst. **54**, 358–367 (1991)

31.49 A. Nüchter, J. Elseberg, P. Schneider, D. Paulus: Study of parameterizations for the rigid body transformations of the scan registration problem, J. Comput. Vis. Image Underst. **114**(8), 963–980 (2010)

31.50 M.A. Fischler, R.C. Bolles: Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography, Comm. ACM **24**(6), 381–395 (1981)

31.51 N.J. Mitra, A. Nguyen: Estimating surface normals in noisy point cloud data, Proc. Symp. Comput. Geom. (SCG) (2003) pp. 322–328

31.52 H. Badino, D. Huber, Y. Park, T. Kanade: Fast and accurate computation of surface normals from

Avidbots Ex1016
Page 842 of 2154

range images, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (2011) pp. 3084–3091
31.53 D. Huber: Automatic Three-Dimensional Modeling from Reality, Ph.D. Thesis (Robotics Institute, Carnegie Mellon University, Pittsburg 2002)
31.54 R.B. Rusu: Semantic 3D Object Maps for Everyday Manipulation in Human Living Environments, Dissertation (TU Munich, Munich 2009)
31.55 Point Cloud Library (PCL): http://www.pointclouds.org (2015)
31.56 J. Böhm, S. Becker: Automatic marker-free registration of terrestrial laser scans using reflectance features, Proc. 8th Conf. Opt. 3D Meas. Tech. (2007) pp. 338–344
31.57 N. Engelhard, F. Endres, J. Hess, J. Sturm, W. Burgard: Real-time 3D visual SLAM with a hand-held camera, Proc. RGB-D Workshop 3D Percept. Robotics at Eur. Robotics Forum (2011)
31.58 R. Schnabel, R. Wahl, R. Klein: Efficient RANSAC for point-cloud shape detection, Computer Graph. Forum (2007)
31.59 A. Hoover, G. Jean-Baptiste, X. Jiang, P.J. Flynn, H. Bunke, D. Goldgof, K. Bowyer, D. Eggert, A. Fitzgibbon, R. Fisher: An experimental comparison of range segmentation algorithms, IEEE Trans. Pattern Anal. Mach. Intell. **18**(7), 673–689 (1996)
31.60 U. Bauer, K. Polthier: Detection of planar regions in volume data for topology optimization, Proc. 5th Int. Conf. Adv. Geom. Model. Process. (2008)
31.61 P.V.C. Hough: Method and means for recognizing complex patterns, Patent US 306 9654 (1962)
31.62 D. Borrmann, J. Elseberg, A. Nüchter, K. Lingemann: The 3D Hough transform for plane detection in point clouds – A review and a new accumulator design, J. 3D Res. **2**(2), 1–13 (2011)
31.63 R. Lakaemper, L.J. Latecki: Extended EM for planar approximation of 3D data, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (2006)
31.64 O. Wulf, K.O. Arras, H.I. Christensen, B.A. Wagner: 2D Mapping of cluttered indoor environments by means of 3D perception, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (2004) pp. 4204–4209
31.65 G. Yu, M. Grossberg, G. Wolberg, I. Stamos: Think globally, cluster locally:A unified framework for range segmentation, Proc. 4th Int. Symp. 3D Data Process. Vis. Transm. (2008)
31.66 W.E. Lorensen, H.E. Cline: Marching Cubes: A high resolution 3D surface construction algorithm, Computer Graph. **21**(4), 163–169 (1987)
31.67 M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, C.T. Silva: Computing and rendering point set surfaces, IEEE Trans. Vis. Comput. Graph. **9**(1), 3–15 (2003)
31.68 H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, W. Stuetzle: Surface reconstruction from unorganized points, Comput. Graph. **26**(2), 71–78 (1992)
31.69 S. Melax: A Simple, fast and effective polygon reduction algorithm, Game Dev. **5**(11), 44–49 (1998)
31.70 M. Garland, P. Heckbert: Surface simplification using quadric error metrics, Proc. SIGGRAPH (1997)
31.71 S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison, A. Fitzgibbon: KinectFusion: Real-time 3D reconstruction and interaction using a moving depth camera, ACM Symp. User Interface Softw. Technol. (2011)
31.72 J.D. Foley, A. van Dam, S.K. Feiner, J.F. Hughes: *Computer Graphics: Principles and Practice*, 2nd edn. (Addison-Wesley, Reading 1996)
31.73 J. Elseberg, D. Borrmann, A. Nüchter: One billion points in the cloud – An octree for efficient processing of 3D laser scans, ISPRS J. Photogramm. Remote Sens. **76**, 76–88 (2013)
31.74 A. Hornung, K.M. Wurm, M. Bennewitz, C. Stachniss, W. Burgard: OctoMap: An efficient probabilistic 3D mapping framework based on octrees, Auton. Robots **34**(3), 189–206 (2013)
31.75 F. Lu, E. Milios: Globally consistent range scan alignment for environment mapping, Auton. Robots **4**, 333–349 (1997)
31.76 K. Konolige: Large-scale map-making, Proc. Natl. Conf. Artif. Intell. (AAAI) (2004) pp. 457–463
31.77 A. Kelly, R. Unnikrishnan: Efficient construction of globally consistent ladar maps using pose network topology and nonlinear programming, Proc. Int. Symp. Robotics Res. (2003)
31.78 D. Borrmann, J. Elseberg, K. Lingemann, A. Nüchter, J. Hertzberg: Globally consistent 3d mapping with scan matching, J. Robotics Auton. Syst. **56**(2), 130–142 (2008)
31.79 E. Grimson, T. Lozano-Pérez, D.P. Huttenlocher: *Object Recognition by Computer: The Role of Geometric Constraints* (MIT Press, Cambridge 1990)
31.80 Z. Zhang: Parameter estimation techniques: a tutorial with application to conic fitting, Image Vis. Comput., Vol. 15 (1997) pp. 59–76
31.81 P. Benko, G. Kos, T. Varady, L. Andor, R.R. Martin: Constrained fitting in reverse engineering, Computer Aided Geom. Des. **19**, 173–205 (2002)
31.82 M. Levoy, K. Pulli, B. Curless, S. Rusinkiewicz, D. Koller, L. Pereira, M. Ginzton, S. Anderson, J. Davis, J. Ginsberg, J. Shade, D. Fulk: The digital Michelangelo project: 3D scanning of large statues, Proc. 27th Conf. Computer Graph. Interact. Tech. (SIGGRAPH) (2000) pp. 131–144
31.83 I. Stamos, P. Allen: 3-D model construction using range and image data, Proc. IEEE Conf. Computer Vis. Pattern Recognit., Vol. 1 (2000) pp. 531–536
31.84 S. Thrun, W. Burgard, D. Fox: A real-time algorithm for mobile robot mapping with applications to multi-robot and 3D mapping, Proc. IEEE Inf. Conf. Robotics Autom. (2000) pp. 321–328
31.85 M. Bosse, R. Zlot, P. Flick: Zebedee: Design of a spring-mounted 3-D range sensor with application to mobile mapping, IEEE Trans. Robotics **28**(5), 1104–1119 (2012)
31.86 The DARPA Grand Challenge: http://archive.darpa.mil/grandchallenge05/gcorg/index.html (2015)
31.87 S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, K. Lau, C. Oakley, M. Palatucci, V. Pratt, P. Stang, S. Strohband,

Avidbots Ex1016
Page 843 of 2154

C. Dupont, L.-E. Jendrossek, C. Koelen, C. Markey, C. Rummel, J. van Niekerk, E. Jensen, P. Alessandrini, G. Bradski, B. Davies, S. Ettinger, A. Kaehler, A. Nefian, P. Mahoney: Stanley: The robot that won the DARPA grand challenge, J. Field Robot. **23**(9), 661–692 (2006)

31.88 R. Triebel, P. Pfaff, W. Burgard: Multi-level surface maps for outdoor terrain mapping and loop closing, Proc. IEEE Int. Conf. Intel. Robots Syst. (IROS) (2006)

31.89 C. Eveland, K. Konolige, R. Bolles: Background modeling for segmentation of video-rate stereo sequences, Proc. Int. Conf. Computer Vis. Pattern Recog. (1998) pp. 266–271

31.90 M. Agrawal, K. Konolige, L. Iocchi: Real-time detection of independent motion using stereo, IEEE Workshop Motion (2005) pp. 207–214

31.91 K. Konolige, M. Agrawal, R.C. Bolles, C. Cowan, M. Fischler, B. Gerkey: Outdoor mapping and Navigation using stereo vision, Intl. Symp. Exp. Robotics (ISER) (2006)

31.92 M. Happold, M. Ollis, N. Johnson: Enhancing supervised terrain classification with predictive unsupervised learning, Robotics: Sci. Syst. Phila. (2006)

31.93 J. Lalonde, N. Vandapel, D. Huber, M. Hebert: Natural terrain classification using three-dimensional ladar data for ground robot mobility, J. Field Robotics **23**(10), 839–862 (2006)

31.94 R. Manduchi, A. Castano, A. Talukder, L. Matthies: Obstacle detection and terrain classification for autonomous off-road navigation, Auton. Robots **18**, 81–102 (2005)

31.95 J.-F. Lalonde, N. Vandapel, M. Hebert: Data structure for efficient processing in 3-D, Robotics: Sci. Syst. (2005)

31.96 A. Kelly, A. Stentz, O. Amidi, M. Bode, D. Bradley, A. Diaz-Calderon, M. Happold, H. Herman, R. Mandelbaum, T. Pilarski, P. Rander, S. Thayer, N. Vallidis, R. Warner: Toward reliable off road autonomous vehicles operating in challenging environments, Int. J. Robotics Res. **25**(5/6), 449–483 (2006)

31.97 P. Bellutta, R. Manduchi, L. Matthies, K. Owens, A. Rankin: Terrain perception for Demo III, Proc. IEEE Intell. Veh. Conf. (2000) pp. 326–331

31.98 KARTO: Software for robots on the move, http://www.kartorobotics.com (2015)

31.99 The Stanford Artificial Intelligence Robot: http://www.cs.stanford.edu/group/stair (2015)

31.100 Perception for Humanoid Robots: https://www.ri.cmu.edu/research_project_detail.html?project_id=595 (2015)

Avidbots Ex1016
Page 844 of 2154

Multimedia Contents

32. 3-D Vision for Navigation and Grasping

Danica Kragic, Kostas Daniilidis

In this chapter, we describe algorithms for three-dimensional (3-D) vision that help robots accomplish navigation and grasping. To model cameras, we start with the basics of perspective projection and distortion due to lenses. This projection from a 3-D world to a two-dimensional (2-D) image can be inverted only by using information from the world or multiple 2-D views. If we know the 3-D model of an object or the location of 3-D landmarks, we can solve the pose estimation problem from one view. When two views are available, we can compute the 3-D motion and triangulate to reconstruct the world up to a scale factor. When multiple views are given either as sparse viewpoints or a continuous incoming video, then the robot path can be computer and point tracks can yield a sparse 3-D representation of the world. In order to grasp objects, we can estimate 3-D pose of the end effector or 3-D coordinates of the graspable points on the object.

32.1 Geometric Vision 812
32.1.1 Calibration 812
32.1.2 Pose Estimation or PnP 813
32.1.3 Triangulation 814
32.1.4 Moving Stereo 815
32.1.5 Structure from Motion 815
32.1.6 Multiple Views SfM 818
32.2 3-D Vision for Grasping 820
32.3 Conclusion and Further Reading 822
Video-References 822
References 822

With the rapid progress and cost reduction in digital imaging, cameras became the standard and probably the cheapest sensor on a robot. Unlike positioning (global positioning system – GPS), inertial (IMU), and distance sensors (sonar, laser, infrared) cameras produce the highest bandwidth of data. Exploiting information useful for a robot from such a bit stream is less explicit than in case of GPS or a laser scanner but semantically richer. In the years since the first edition of the handbook, we had significant advances in hardware and algorithms. RGB-D sensors like the Primesense Kinect enabled a new generation of full model reconstruction systems [32.1] with an arbitrary camera motion. Google's project Tango [32.2] established the state of the art in visual odometry using the latest fusion methods between visual and inertial data ([32.3] and VIDEO 120). 3-D modeling became a commodity software (see, for example, 123D Catch App from Autodesk) and the widely used open source Bundler ([32.4] VIDEO 121) has been possible by advances in wide baseline matching and bundle adjustment. Methods for wide baseline matching have been proposed for several variations of pose and structure from motion [32.5]. Last, the problem of local minima for nonminimal overconstrained solvers has been addressed by a group of method using Branch and Bound global optimization of a sum of fractions subject to convex constraints [32.6] or an L_∞-norm of the error function [32.7].

Let us consider the two main robot perception domains: navigation and grasping. Assume for example the scenario that a robot vehicle is given the task of going from place A to place B given as instruction only intermediate visual landmarks and/or GPS waypoints. The robot starts at A and has to decide where is a drivable path. Such a decision can be accomplished through

the detection of obstacles from at least two images by estimating a depth or occupancy map with a *stereo* algorithm. While driving, the robot wants to estimate its trajectory which can be accomplished with a *matching* and *structure from motion* algorithm. The result of the trajectory can be used to build a lay out of the environment through dens matching and *triangulation* which in turn can be used as a reference for a subsequent *pose estimation*. At each time instance the robot has to parse the surrounding environment for risks like pedestrians, or for objects it is searching for like a trash-can. It has to become aware of *loop closing* or a reentry if the robot has been kidnaped or blind for a while. This can be accomplished through *object and scene recognition* yielding the *what* and *where* of objects around the robot. In an extreme scenario, a vehicle can be left to explore a city and build a semantic 3-D map as well as a trajectory of all places it visited, the ultimate *visual simultaneous localization and semantic mapping* problem. In the case of grasping, the robot detects an object given a learnt representation, and subsequently, it has to estimate the *pose* of the object and in some cases its shape by *triangulation*. When a camera is not mounted on an end-effector, the *absolute orientation* between the hand the object has to be found.

In the next section we will present the geometric foundations for 3-D vision and in the last section we describe approaches for grasping.

32.1 Geometric Vision

Let us start by introducing the projection of the world to an image plane. Assume that a point in the world (X, Y, Z) has coordinates (X_{ci}, Y_{ci}, Z_{ci}) with respect to the coordinate system of a camera c_i related to each other by the following transformation

$$\begin{pmatrix} X_{ci} \\ Y_{ci} \\ Z_{ci} \end{pmatrix} = \mathbf{R}_i \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} + \boldsymbol{T}_i \,, \tag{32.1}$$

where $\mathbf{R}_i$ is a rotation matrix whose columns are the world axes with respect to the camera. The translation vector $\boldsymbol{T}_i$ is starting from the origin of the camera and ending at the origin of the world coordinate system. The rotation matrix is orthogonal $\mathbf{R}^{\mathrm{T}}\mathbf{R} = \mathbf{1}$ with determinant one. We assume that the center of projection is the origin of the coordinate system and that the optical axis is the Z_{ci} axis of the camera. If we assume that the image plane is the plane $Z_{ci} = 1$ then the image coordinates (x_i, y_i) read

$$x_i = \frac{X_{ci}}{Z_{ci}} \,, \quad y_i = \frac{Y_{ci}}{Z_{ci}} \,. \tag{32.2}$$

In practice, what we measure are the pixel coordinates (u_i, v_i) in the image which are related to image coordinates (x_i, y_i) with the affine transformation

$$u_i = f\alpha x_i + \beta y_i + c_u \,, \quad v_i = f y_i + c_v \,, \tag{32.3}$$

where f is the distance of the image plane to the projection center measured in pixels. It is also called focal length, because they are considered approximately equal. The aspect ratio α is a scaling induced by nonsquare sensor cells or different sampling rates horizontally and vertically. The skew factor β accounts for a shearing induced by a nonperfectly frontal image plane. The image center c_u, c_v is the point of intersection of the image plane with the optical axis called the image center. These five parameters are called intrinsic parameters and the process of recovering them is called intrinsic calibration. Upon recovering them we can talk about a calibrated system and we can work with the image coordinates (x_i, y_i) instead of the pixel coordinates (u_i, v_i). In many vision systems in particular on mobile robots, wide-angle lenses introduce a radial distortion around the image center which can be modelled polynomially

$$\begin{aligned} x_i^{\mathrm{dist}} &= x_i(1 + k_1 r + k_2 r^2 + k_3 r^3 + \ldots) \\ y_i^{\mathrm{dist}} &= y_i(1 + k_1 r + k_2 r^2 + k_3 r^3 + \ldots) \\ \text{where} \quad r^2 &= x_i^2 + y_i^2 \,, \end{aligned}$$

where we temporarily assumed that the image center is at (0,0). The image coordinates (x_i, y_i) in ((32.3)) have to be replaced with the distorted coordinates $(x^{\mathrm{dist}}, y^{\mathrm{dist}})$.

32.1.1 Calibration

Recovering the intrinsic parameters when we can make multiple views of a reference pattern like a checker board without variation of the intrinsic parameters has become a standard procedure using tools like the MATLAB calibration toolbox or *Zhang*'s OpenCV calibration function [32.8]. When intrinsics like the focal length vary during operation and viewing reference patterns is not practically feasible, we rely on the state of the art method by *Pollefeys* et al. [32.9, 10]. When all intrinsic are unknown on the Kruppa equations and

several stratified self-calibration approaches [32.11, 12] which require at least three views. Apart radial distortion, the projection relations shown above can be summarized in matrix form. By denoting

$$\boldsymbol{u}_i = (u_i, v_i, 1)$$

and

$$\boldsymbol{X} = (X, Y, Z, 1)$$

we obtain

$$\lambda_i \boldsymbol{u}_i = \mathbf{K}_i \begin{pmatrix} \mathbf{R}_i & \boldsymbol{T}_i \end{pmatrix} \boldsymbol{X} = \mathbf{P}\boldsymbol{X} \,, \tag{32.4}$$

where $\lambda_i = Z_{ci}$ is the depth of point $\boldsymbol{X}$ in camera coordinates and $\mathbf{P}$ is the 3×4 projection matrix. The depth λ_i which can be eliminated to obtain two equations relating the world to the pixel coordinates.

32.1.2 Pose Estimation or PnP

When we have landmarks in the world with known positions $\boldsymbol{X}$ and we can measure their projections, the problem of recovering the unknown rotation and translation in the calibrated case is called pose estimation or the Prespective-n-Point problem (PnP). Of course, it presumes the identification of the world points in the image. In robotics, the pose estimation is a variant of the localization problem in a known environment. When grasping objects of known shape PnP yields the target pose for an end-effector module the grasping point positions. We assume that a camera is calibrated and that measurements of N points are given in world coordinates $\boldsymbol{X}_{j=1..N}$ and calibrated image coordinates $\boldsymbol{x}_{j=1..N}$. Let us assume two scene points and denote the known angle between their projections $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ as δ_{12} (Fig. 32.1). Let us denote the squared distance $\|\boldsymbol{X}_i - \boldsymbol{X}_j\|^2$ with d_{ij}^2 and the lengths of $\boldsymbol{X}_j$ with d_j^2. Then cosine law reads

$$d_i^2 + d_j^2 - 2 d_i d_j \cos\delta_{ij} = d_{ij}^2 \,. \tag{32.5}$$

If we can recover d_i and d_j the rest will be an absolute orientation problem

$$\mathrm{d}_j \boldsymbol{x}_j = \mathbf{R}\boldsymbol{X}_j + \boldsymbol{T} \tag{32.6}$$

to recover translation and rotation between camera and world coordinate system.

Minimal Solution

The cosine law has two unknowns d_1 and d_2 so with three points we should be able to solve for the pose estimation problem. Indeed, three points yield a system of three quadratic equations in three unknowns, so it will have a maximum of eight solutions.

We follow here the analysis of the classic solution in [32.13] and set $d_2 = u d_1$ and $d_3 = v d_1$ and solve all three equations for d_1

$$d_1^2 = \frac{d_{23}^2}{u^2 + v^2 - 2uv\cos\delta_{23}} \,,$$
$$d_1^2 = \frac{d_{13}^2}{1 + v^2 - 2v\cos\delta_{13}} \,,$$
$$d_1^2 = \frac{d_{12}^2}{u^2 + 1 - 2u\cos\delta_{12}} \,,$$

which is equivalent to two quadratic equations in u and v

$$d_{12}^2(1 + v^2 - 2v\cos\delta_{13}) = d_{13}^2(u^2 + 1 - 2u\cos\delta_{12}) \,, \tag{32.7}$$
$$d_{13}^2(u^2 + v^2 - 2uv\cos\delta_{23}) = d_{23}^2(1 + v^2 - 2v\cos\delta_{13}) \,. \tag{32.8}$$

Solving (32.8) for u^2 and substituting in (32.7) allows solving E1 for u because u appears linearly. Substituting u back in (32.8) yields a quartic in v which can have as many as four real roots. For each v we obtain two roots for u through any of the quadratic equations yielding a maximum of eight solutions [32.13, 14]. Popular pose estimation algorithms are based either on an iterative method [32.15, 16] or linear versions using auxiliary unknowns of higher dimension [32.17, 18].

A more recent method [32.19] for n world points expresses 3-D points as the barycentric coordinates with

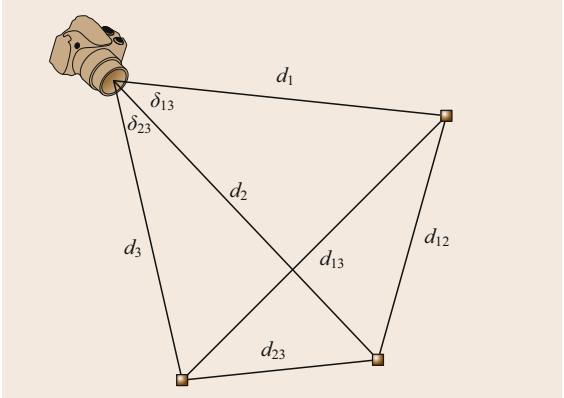


Fig. 32.1 Pose estimation problem: A camera seeing 3 points at unknown distances d_1, d_2, and d_3 with known angles between the rays and known point distances d_{12}, d_{13}, d_{23}

Avidbots Ex1016
Page 847 of 2154

respect to four virtual control points

$$\boldsymbol{X}_i = \sum_{j=1}^{4} \alpha_{ij} \boldsymbol{C}_j \,, \quad \text{where} \quad \sum_{j=1}^{4} \alpha_{ij} = 1 \,.$$

A rigid transformation to the camera coordinate system leaves the barycentric coordinates invariant and a perspective projection yields

$$\lambda_i \boldsymbol{x}_i = \sum_{j=1}^{4} \alpha_{ij} \left(X_{\mathrm{c}i}, Y_{\mathrm{c}i}, Z_{\mathrm{c}i}\right)^{\mathrm{T}} \,.$$

Eliminating λ_i yields two linear equations for each point

$$\sum_{j=1}^{4} \alpha_{ij} C_{x_{\mathrm{c}j}} = \alpha_{ij} x_i C_{z_{\mathrm{c}j}}$$

$$\sum_{j=1}^{4} \alpha_{ij} C_{y_{\mathrm{c}j}} = \alpha_{ij} y_i C_{z_{\mathrm{c}j}}$$

with the coordinate triples of the control points in the camera frame being the 12 unknowns. This is a linear homogeneous system with the solution being the nullspace of a $2n \times 12$ matrix. The unknown control points are found up to a scale factor which is easily fixed because we know the inter point distances. The pose is found from absolute orientation between control points in the camera and the world frame. This yields a very efficient solution for $n \geq 6$ points but leaves you with the initial choice of the control points as a factor affecting the solution.

In case that $n \geq 4$ points lie on a plane we can compute the homography H between the world and the camera plane [32.8]. Assuming $Z = 0$ is the world plane the homography reads

$$\begin{pmatrix} u \\ v \\ w \end{pmatrix} \approx \underbrace{K \begin{pmatrix} r_1 & r_2 & T \end{pmatrix}}_{H} \begin{pmatrix} X \\ Y \\ W \end{pmatrix} \,,$$

where $r_{1,2}$ are the first two columns of the rotation matrix and $\approx$ denotes the projective equivalence, namely, for any two points $\boldsymbol{p}$ and $\boldsymbol{p}'$ in the projective plane $\boldsymbol{p} \approx \boldsymbol{p}'$ iff $\boldsymbol{p} = \lambda \boldsymbol{p}'$ for real $\lambda \neq 0$. Hence the first two columns of $K^{-1}H$

$$K^{-1}H = \begin{pmatrix} h'_1 & h'_2 & h'_3 \end{pmatrix}$$

have to be orthogonal. We seek thus an orthogonal matrix $\mathbf{R}$ that is the closest to $\begin{pmatrix} h'_1 & h'_2 & h'_1 \times h'_2 \end{pmatrix}$

$$\arg\min_{R \in \mathrm{SO}(3)} \| \mathbf{R} - \begin{pmatrix} h'_1 & h'_2 & h'_1 \times h'_2 \end{pmatrix} \|_F^2 \,.$$

If the singular value decomposition (SVD) of

$$\begin{pmatrix} h'_1 & h'_2 & h'_1 \times h'_2 \end{pmatrix} = USV^{\mathrm{T}} \,,$$

then the solution is [32.20]

$$\mathbf{R} = U \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \det(UV^{\mathrm{T}}) \end{pmatrix} V^{\mathrm{T}} \,. \tag{32.9}$$

The diagonal matrix is a projection from the orthogonal group O(3) to the special orthogonal group SO(3).

Last, we present a method [32.21] for n points that computes all local minima of the over constrained PnP problem. This involves solving the first derivatives explicitly with respect to the pose unknowns. To achieve this, following observation allows the elimination of the depths λ and the translation. Rigid transformation $\lambda \boldsymbol{x} = \mathbf{R}\boldsymbol{X} + \boldsymbol{T}$ can be written for n points as a linear system for $\lambda_{j=1..n}$ and the translation $\boldsymbol{T}$

$$\begin{pmatrix} \boldsymbol{x}_1 & & & -I \\ & \ddots & & \vdots \\ & & \boldsymbol{x}_n & -I \end{pmatrix} \begin{pmatrix} \lambda_1 \\ \vdots \\ \lambda_n \\ \boldsymbol{T} \end{pmatrix} = \begin{pmatrix} \mathbf{R}\boldsymbol{X}_1 \\ \vdots \\ \mathbf{R}\boldsymbol{X}_n \end{pmatrix} \,.$$

We can solve for the unknown depths-translation vector and back substitute it into a least squares minimization problem with respect to rotation parameters. It turns out that if we use the three Rodriguez parameters as rotation parametrization the necessary conditions for an extremum (vanishing derivatives) turn out to be three cubic equations [32.21]. Last we would like to point out to the reader that a nonlinear function of the rotation matrix can also be solved as an optimization problem on the Lie-group SO(3) [32.22–24] for the case of line correspondences.

32.1.3 Triangulation

When we know both the intrinsics and extrinsics or their summarization in matrix $\mathbf{P}$ and we measure a point we cannot recover its depth from just one camera position. Assuming that we have the projection of the same point $\boldsymbol{X}$ in two cameras

$$\lambda_1 \boldsymbol{u}_1 = \mathbf{P}_1 \begin{pmatrix} \boldsymbol{X} \\ 1 \end{pmatrix} \,,$$

$$\lambda_2 \boldsymbol{u}_2 = \mathbf{P}_2 \begin{pmatrix} \boldsymbol{X} \\ 1 \end{pmatrix} \,, \tag{32.10}$$

with known projection matrices $\mathbf{P}_1$ and $\mathbf{P}_2$ we can recover the position $\boldsymbol{X}$ in space, a process well known

as triangulation. Observe that we can achieve triangulation without decomposing the projection matrices into intrinsic and extrinsic parameters, we neeed though to remove the distortion in order to write them as above.

Having correspondences of the same point in two cameras with known projection matrices $\mathbf{P}_\mathrm{l}$ and $\mathbf{P}_\mathrm{r}$ we can solve the two projection equations for the world point $\boldsymbol{X}$. It is worth noting that each point provides two independent equations so that triangulation becomes an overconstrained problem for two views. This is not a contradiction since two rays do not intersect in general in space unless they satisfy the epipolar constraint as presented in the next paragraph. The following matrix in the left hand side has in general rank 4 unless the epipolar constraint is satisfied in which case it has rank 3.

$$\begin{pmatrix} x\mathbf{P}_\mathrm{l}(3,:)-\mathbf{P}_\mathrm{l}(1,:) \\ y\mathbf{P}_\mathrm{l}(3,:)-\mathbf{P}_\mathrm{l}(2,:) \\ x\mathbf{P}_\mathrm{r}(3,:)-\mathbf{P}_\mathrm{r}(1,:) \\ y\mathbf{P}_\mathrm{r}(3,:)-\mathbf{P}_\mathrm{r}(2,:) \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = \mathbf{0} \,, \tag{32.11}$$

where $\mathbf{P}(i,:)$ means the i-th row of matrix $\mathbf{P}$.

Obviously, the homogeneous system above can be transformed into an inhomogeneous linear system with unknowns (X, Y, Z). Otherwise it can be solved by finding the vector closest to the null-space of the 4×4 matrix above using SVD. A thorough treatment of triangulation is the classic [32.25].

32.1.4 Moving Stereo

Imagine now that a rigid stereo system consisting of cameras c_l (left) and c_r (right)

$$\boldsymbol{u}_{\mathrm{l}i} \approx \mathbf{P}_\mathrm{l}\boldsymbol{X}_i \,, \tag{32.12}$$
$$\boldsymbol{u}_{\mathrm{r}i} \approx \mathbf{P}_\mathrm{r}\boldsymbol{X}_i \,, \tag{32.13}$$

is attached to a moving robot and observe this system at two time instances

$$\boldsymbol{X}_0 = \mathbf{R}_1\boldsymbol{X}_1 + \boldsymbol{T}_1 \,, \tag{32.14}$$

where $\boldsymbol{X}_0$ are point coordinates with respect to the world coordinate system, usually assumed aligned with one of the camera instances, and $\boldsymbol{X}_1$ are the coordinates of the same point with respect to the camera rig, after a motion $(\mathbf{R}_1, \boldsymbol{T}_1)$. To estimate the motion of the rig, we have to solve two correspondence problems, first, between left and right image, and second, between left (or right) at the first time instance and left (or right, respectively) at the second time instance. The left to right correspondence enable the solution of the triangulation problem at each time instance. Motion can be obtained then by solving equations (32.14) for $(\mathbf{R}_1, \boldsymbol{T}_1)$, a problem called absolute orientation. Alternatively one can avoid the second triangulation and solve the pose estimation problem between triangulated points in 3-D and points in the left image only. The most popular visual odometry system today is `libviso` [32.26] and is based on a moving stereo rig (VIDEO 122).

Absolute Orientation

The treatment for moving stereo will be short and the reader is referred to a similar treatment in the chapter about range sensing. We assume that correspondences between two time instances have been established based on tracking in the images so that we can formulate equations of the form

$$\boldsymbol{X}_2 = \mathbf{R}\boldsymbol{X}_1 + \boldsymbol{T} \,.$$

The standard way [32.20, 27] to solve this problem is to eliminate the translation by subtracting the centroids yielding

$$\boldsymbol{X}_2 - \overline{\boldsymbol{X}_2} = \mathbf{R}(\boldsymbol{X}_1 - \overline{\boldsymbol{X}_1}) \,.$$

We need at least three points in total to obtain at least two noncollinear mean-free $\boldsymbol{X}-\bar{\boldsymbol{X}}$ vectors. If we concatenate the mean free vectors for n points into an $n\times3$ matrix $A_{1,2}$ we can formulate the following minimization of the Frobenius norm

$$\min_{R\in\mathrm{SO}(3)} \|A_2 - \mathbf{R}A_1\|_F \,,$$

which is known as the Procrustes problem. It can be shown [32.20] that the solution is obtained through SVD as in (32.9) where U, V are obtained from the singular value decomposition

$$A_2A_1^\mathrm{T} = USV^\mathrm{T} \,.$$

Solutions are usually obtained with RANSAC by sampling triples of points and verification with the Procrustes method.

32.1.5 Structure from Motion

Relax now the assumption that projection matrices are known and remain with measuring and matching corresponding points $\boldsymbol{u}_1$ and $\boldsymbol{u}_2$. This is the well known structure from motion problem or more precisely structure and 3-D-motion from 2-D motion. In photogrammetry, it is well known as relative orientation problem. Even after eliminating the λ's from equations (32.12)

Avidbots Ex1016
Page 849 of 2154

or by writing them in projective equivalence form

$$\boldsymbol{u}_1 \approx \mathbf{P}_1 \begin{pmatrix} \boldsymbol{X} \\ 1 \end{pmatrix} ,$$
$$\boldsymbol{u}_2 \approx \mathbf{P}_2 \begin{pmatrix} \boldsymbol{X} \\ 1 \end{pmatrix} , \quad (32.15)$$

we realize that we if $(\boldsymbol{X}, \mathbf{P}_1, \mathbf{P}_2)$ is a solution than $(\mathbf{H}\boldsymbol{X}, \mathbf{P}_1\mathbf{H}^{-1}, \mathbf{P}_2\mathbf{H}^{-1})$ is a solution, too, where $\mathbf{H}$ an invertible 4×4 real matrix or in other words a collineation in $\mathbb{P}^3$. Even if we align the world coordinate system with the coordinate system of the first camera, which practice is common

$$\boldsymbol{u}_1 \approx \begin{pmatrix} \boldsymbol{I} & \mathbf{0} \end{pmatrix} \boldsymbol{X} ,$$
$$\boldsymbol{u}_2 \approx \mathbf{P}_2 \boldsymbol{X} , \quad (32.16)$$

we remain with the same ambiguity where $\mathbf{H}$ is of the form

$$\mathbf{H} \approx \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ h_{41} & h_{42} & h_{43} & h_{44} \end{pmatrix} , \quad (32.17)$$

with $h_{44} \neq 0$. This ambiguity is possible when the projection matrices are arbitrary rank 3 real matrices without any constraint on their elements. If we assume that we have calibrated our cameras then the projection matrices depend only on displacements

$$\boldsymbol{u}_1 \approx \begin{pmatrix} \mathbf{I} & \mathbf{0} \end{pmatrix} \boldsymbol{X} ,$$
$$\boldsymbol{u}_2 \approx \begin{pmatrix} \mathbf{R} & \boldsymbol{T} \end{pmatrix} \boldsymbol{X} , \quad (32.18)$$

and the only remaining ambiguity is the scale ambiguity where $\mathbf{H}$ looks like an identity matrix except $h_{44} = s \neq 1$ being the scale factor. In other words if $(\mathbf{R}, \boldsymbol{T}, \boldsymbol{X})$ is a solution then $(\mathbf{R}, s\boldsymbol{T}, 1/s\boldsymbol{X})$ is a solution, too. These remarks generalize in multiple views. Because, in robotics the $(\mathbf{R}, \boldsymbol{T})$ matrices correspond to location and $\boldsymbol{X}$ to mapping of the environment, the problem has the more proper term SLAM: Simultaneous localization and mapping. However, because the term SLAM has been used with a variety of sensors like sonar and laser range scanners, the term monocular SLAM is better suited to describe structure from motion from multiple views [32.28].

Epipolar Geometry

This is probably one of the most studied problems in computer vision. We constrain ourselves to the calibrated case which is most relevant to robotics applications. The necessary and sufficient condition for the intersection of the two rays $\mathbf{R}\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ is that the two rays are coplanar with the baseline $\boldsymbol{T}$

$$\boldsymbol{x}_2^{\mathrm{T}}(\boldsymbol{T} \times \mathbf{R}\boldsymbol{x}_1) = 0 , \quad (32.19)$$

which is the epipolar constraint (Fig. 32.2). To avoid the scale ambiguity we assume that $\boldsymbol{T}$ is a unit vector. We proceed by summarizing the unknowns into one matrix

$$\mathbf{E} = \hat{\mathbf{T}}\mathbf{R} \quad (32.20)$$

where $\hat{\boldsymbol{T}}$ is the 3×3 skew-symmetric matrix to the vector $\boldsymbol{T}$. The $\mathbf{E}$ matrix is called the essential matrix. The epipolar constraint reads then

$$\boldsymbol{x}_2^{\mathrm{T}}\mathbf{E}\boldsymbol{x}_1 = 0 , \quad (32.21)$$

which is the equation of a line in the $\boldsymbol{x}_2$ plane with coefficients $E\boldsymbol{x}_1$ or a coefficient of a line in the $\boldsymbol{x}_1$ plane with coefficients $E^{\mathrm{T}}\boldsymbol{x}_2$. These lines are called epipolar and form pencils whose centers are the epipoles $\boldsymbol{e}_1$ and $\boldsymbol{e}_2$, in the first and second image plane respectively. The epipoles are the intersections of the baseline with the two image planes, hence $\boldsymbol{e}_2 \approx \boldsymbol{T}$ and $\boldsymbol{e}_1 \approx -\mathbf{R}^{\mathrm{T}}\boldsymbol{T}$. Looking at the equations of the epipolar lines we can immediately infer that $E^{\mathrm{T}}\boldsymbol{e}_1 = 0$ and $E\boldsymbol{e}_2 = 0$.

The set of all essential matrices

$$\mathcal{E} = \left\{ \mathbf{E} \in \mathbb{R}^{3\times3} \mid \mathbf{E} = \widehat{\boldsymbol{T}}\mathbf{R}, \right.$$
$$\left. \text{where } \boldsymbol{T} \in \mathbb{S}^2 \text{ and } \mathbf{R} \in \mathrm{SO}(3) \right\}$$

has been characterized as a manifold of dimension 5 [32.29]. It has been proven [32.30] that

Proposition 32.1
A matrix $\mathbf{E} \in \mathbb{R}^{3\times3}$ is essential if and only if it has two singular values equal to each other and third singular value equal zero.

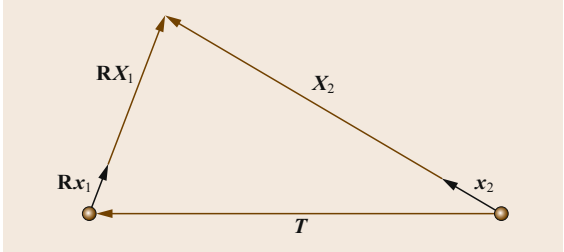


Fig. 32.2 A point is perspectively projected to calibrated image vectors $\mathbf{R}\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ which are coplanar with baseline $\boldsymbol{T}$

We present here Nister's method [32.31] for recovering an essential matrix from five point correspondences and which gained in popularity because of its suitability for RANSAC methods.

Minimal Case

We expand the epipolar constraint in terms of homogeneous coordinates $\boldsymbol{x}_1 = (x_1, y_1, z_1)$ and $\boldsymbol{x}_2 = (x_2, y_2, z_2)$ (when the points are not at infinity $z_i = 1$) and obtain

$$\begin{pmatrix} x_1\boldsymbol{x}_2^{\mathrm{T}} & y_1\boldsymbol{x}_2^{\mathrm{T}} & z_1\boldsymbol{x}_3^{\mathrm{T}} \end{pmatrix} E_{\mathrm{s}} = 0\,, \tag{32.22}$$

where E_{s} is the raw by raw stacked version of matrix $\mathbf{E}$. When we use only five point correspondences the resulting linear homogeneous system will have as a solution any vector in the four dimensional kernel of the data matrix

$$E_{\mathrm{s}} = \lambda_1\boldsymbol{u}_1 + \lambda_2\boldsymbol{u}_2 + \lambda_3\boldsymbol{u}_3 + \lambda_4\boldsymbol{u}_4\,. \tag{32.23}$$

At this point we want the matrix $\mathbf{E}$ resulting from E_{s} to be an essential matrix satisfying Proposition 32.1. It has been proven [32.30] that

Proposition 32.2

A matrix $\mathbf{E} \in \mathbb{R}^{3\times3}$ is essential if and only if

$$\mathbf{E}\mathbf{E}^{\mathrm{T}}\mathbf{E} = \frac{1}{2}\mathrm{trace}(\mathbf{E}\mathbf{E}^{\mathrm{T}})\mathbf{E}\,. \tag{32.24}$$

Though the $\det(\mathbf{E}) = 0$ constraint can be inferred from (32.24) we are still going to use it together with (32.24) to obtain ten cubic equations in the elements of $\mathbf{E}$. As described in [32.31], one can obtain a tenth degree polynomial in λ_4. The number of real roots of this polynomial are computed with a Sturm sequence. There is no proof beyond physical plausibility of the existence of at least one solution that a real root will exist at all. Several alternative 5-point solvers have been proposed since *Nister*'s paper [32.32–35] and an extensive list including code has been established by Pajdla's group [32.36].

Assuming that we have recovered an essential matrix from point correspondences, the next task is to recover an orthogonal matrix $\mathbf{R}$ and a unit vector translation $\boldsymbol{T}$ from the essential matrix. If $E = U\mathrm{diag}(\sigma, \sigma, 0)V^{\mathrm{T}}$, there are four solutions for the pair $(\widehat{T}, R)$

$$\begin{aligned}
(\widehat{\boldsymbol{T}}_1, R_1) &= (UR_{z,+\pi/2}\Sigma U^{\mathrm{T}}, UR_{z,+\pi/2}^{\mathrm{T}}V^{\mathrm{T}})\,,\\
(\widehat{\boldsymbol{T}}_2, R_2) &= (UR_{z,-\pi/2}\Sigma U^{\mathrm{T}}, UR_{z,-\pi/2}^{\mathrm{T}}V^{\mathrm{T}})\,,\\
(\widehat{\boldsymbol{T}}_1, R_2) &= (UR_{z,+\pi/2}\Sigma U^{\mathrm{T}}, UR_{z,-\pi/2}^{\mathrm{T}}V^{\mathrm{T}})\,,\\
(\widehat{\boldsymbol{T}}_2, R_1) &= (UR_{z,-\pi/2}\Sigma U^{\mathrm{T}}, UR_{z,+\pi/2}^{\mathrm{T}}V^{\mathrm{T}})\,,
\end{aligned}$$

where R_z denotes rotation around the z-axis. The four solutions can be split into two two-fold ambiguities:

- *Mirror ambiguity*: If T is a solution, then $-T$ is a solution, too. There is no way to disambiguate from the epipolar constraint: $\boldsymbol{x}_2^{\mathrm{T}}((-\boldsymbol{T}) \times R\boldsymbol{x}_1) = 0$.
- *Twisted pair ambiguity*: If R is a solution, then also $R_{T,\pi}R$ is a solution. The first image is *twisted* around the baseline 180 degrees.

These ambiguities are resolved by checking if depths of triangulated points are positive.

Critical Ambiguities

The approach with five point correspondences has a finite number of feasible (feasible means that they may produce multiple interpretations of structures in front of the camera) solutions when the points in the scene lie on a plane (a two fold ambiguity) [32.37] or when the points on the scene and the camera centers lie on a double sheet hyperboloid with the additional constraint that the camera centers lie symmetrically to the main generator of the hyperboloid [32.38]. These are inherent ambiguities which hold for any number of point correspondences when one seeks a solution for an exact essential matrix.

When someone is solving the linear least squares system for the essential matrix, a planar scene as well as the case of all points and the camera centers lying on a quadric causes a rank deficiency of the system and thus infinite solutions for $\mathbf{E}$.

Beyond the ambiguous situations, there is a considerable amount of literature regarding instabilities in the two view problem. In particularly, it has been shown [32.37, 39, 40] that a small field of view and insufficient depth variation can cause an indeterminacy in the estimation of the angle between translation and optical axis. An additional small rotation can cause a confounding between translation and rotation [32.41]. Moreover, it has been shown, that there exist local minima close to the global minimum that can fool any iterative scheme [32.42, 43].

3-Point SfM

Minimal solutions based on 5 points are still too slow to be used on mobile platforms where additional information like a reference gravity vector might be obtained from an IMU. We present here a recent solution using a reference direction and only 3 points [32.44].

We are given three image correspondences from calibrated cameras, and a single directional correspondence like the gravity vector or a vanishing point. This problem is equivalent to finding the translation vector $\boldsymbol{t}$ and a rotation angle θ around an arbitrary rotation axis.

Avidbots Ex1016
Page 851 of 2154

Let us choose the arbitrary rotation axis to be $\mathbf{e}_2 = [0,1,0]^{\mathrm{T}}$. After taking the directional constraint into account, from the initial five parameters in the essential matrix, we now only have to estimate three. We can use the axis-angle parameterization of a rotation matrix to rewrite the essential matrix constraint as follows

$$\mathbf{p}_{2i}{}^{\mathrm{T}}\tilde{E}\mathbf{p}_1 = 0\,, \tag{32.25}$$

where

$$\tilde{E} = \widehat{\tilde{\boldsymbol{t}}}(I + \sin\theta\widehat{\mathbf{e}}_2 + (1-\cos\theta)\widehat{\mathbf{e}}_2^2)\,,$$

and $\tilde{\boldsymbol{t}} = (x, y, 1)$.

Each image point correspondence gives us one such equation, for a total of three equations in three unknowns (elements of $\boldsymbol{t}$ and θ). To create a polynomial system, we set $s = \sin\theta$ and $c = \cos\theta$, and add the trigonometric constraint $s^2 + c^2 - 1 = 0$, for a total of four equations in four unknowns. In order to reduce the number of unknowns, we choose the direction of the epipole by assuming that the translation vector $\tilde{\boldsymbol{t}}$ has the form $[x, y, 1]^{\top}$. This means that for each $\tilde{\boldsymbol{t}}$ that we recover, $-\tilde{\boldsymbol{t}}$ will also need to be considered as a possible solution.

Once we substitute for $\tilde{E}$ in (32.25), the resulting system of polynomial equations has the following form

$$\begin{aligned} &a_{i1}xs + a_{i2}xc + a_{i3}ys + a_{i4}yc \\ &+ a_{i5}x - a_{i2}s + a_{i1}c + a_{i6} = 0 \end{aligned} \tag{32.26}$$

for $i = 1, .., 3$, and the equation

$$s^2 + c^2 - 1 = 0\,. \tag{32.27}$$

This polynomial system can be solved in closed form and has up to four solutions. The total number of possible pose matrices arising from our formulation is therefore at most 8, when we take into account the fact that we have to consider the sign ambiguity in translation.

32.1.6 Multiple Views SfM

When we talk about simultaneous localization and mapping we obviously mean over a longer period of time. The question is how do we integrate additional frames in our 3-D motion estimation (localization) process.

To exploit multiple frames we introduce rank constraints [32.45]. We assume that the world coordinate system coincides with the coordinate system of the first frame and that a scene point is projected to $\boldsymbol{x}_i$ in the i-th frame and that its depth with respect to the 1st frame is λ_1

$$\lambda_i\boldsymbol{x}_i = \mathbf{R}_i(\lambda_1\boldsymbol{x}_1) + \boldsymbol{T}_i\,. \tag{32.28}$$

Taking the cross product with $\boldsymbol{x}_i$ and writing it for n frames yields a homogeneous system

$$\begin{pmatrix} \widehat{\boldsymbol{x}_2}\mathbf{R}_2\boldsymbol{x}_1 & \widehat{\boldsymbol{x}_2}\boldsymbol{T}_2 \\ \vdots & \vdots \\ \widehat{\boldsymbol{x}_n}\mathbf{R}_n\boldsymbol{x}_1 & \widehat{\boldsymbol{x}_2}\boldsymbol{T}_n \end{pmatrix} \begin{pmatrix} \lambda_1 \\ 1 \end{pmatrix} = 0\,, \tag{32.29}$$

that has the depth of a point in the first frame as an unknown. The $3n \times 2$ multiple view matrix has to have rank one [32.46], a constraint that infers both the epipolar and the trifocal equations. The least squares solution for the depth can easily be derived as

$$\lambda_1 = -\frac{\sum_{i=1}^{n}(\boldsymbol{x}_i \times \boldsymbol{T}_i)^{\mathrm{T}}(\boldsymbol{x}_i \times \mathbf{R}_i\boldsymbol{x}_1)}{\|\boldsymbol{x}_i \times \mathbf{R}_i\boldsymbol{x}_1\|^2}\,. \tag{32.30}$$

Given a depth for each point we can solve for motion by rearranging the multiple views constraint (32.29) as

$$\begin{pmatrix} \lambda_1^1\boldsymbol{x}_1^{1T} \otimes \hat{\boldsymbol{x}}_i^1 & \hat{\boldsymbol{x}}_i^1 \\ \vdots & \vdots \\ \lambda_1^n\boldsymbol{x}_1^{nT} \otimes \hat{\boldsymbol{x}}_i^n & \hat{\boldsymbol{x}}_i^n \end{pmatrix} \begin{pmatrix} \boldsymbol{R}_i^{\text{stacked}} \\ \boldsymbol{T}_i \end{pmatrix} = 0\,, \tag{32.31}$$

where $\boldsymbol{x}_i^n$ is the n-th image point in the i-th frame and $\mathbf{R}_i, \boldsymbol{T}_i$ is the motion from 1st to the i-th and $\boldsymbol{R}_i^{\text{stacked}}$ is the 12×1 vector of stacked elements of the rotation matrix $\mathbf{R}_i$. Suppose that $\boldsymbol{k}$ is the 12×1 kernel (or closest kernel in a least squares sense) of the $3n \times 12$ matrix in the left hand side obtained through singular value decomposition and let us call $\mathbf{A}$ the 3×3 matrix obtained from the first 9 elements of $\boldsymbol{k}$ and $\boldsymbol{a}$ the vector of elements 10–12. To obtain a rotation matrix we follow the SVD steps in the solution of absolute orientation (32.14) to find the closest orthogonal matrix to an arbitrary invertible matrix.

Bundle Adjustment

On top of such an approach, a bundle adjustment [32.47] minimizes the sum of all deviations between image coordinates and the backprojections of the points to be reconstructed.

$$\arg\min_{R^f, T^f, X_p} \epsilon^{\mathrm{T}} C^{-1} \epsilon\,,$$

minimized with respect to all $6(F-1)$ motions and $3N-1$ structure unknowns, where ϵ is the vector con-

taining all errors

$$\epsilon_p^f = \left(x_p^f - \frac{R_{11}^f X_p + R_{12}^f Y_p + R_{13}^f Z_p + T_x}{R_{31}^f X_p + R_{32}^f Y_p + R_{33}^f Z_p + T_z} \right.$$
$$\left. y_p^f - \frac{R_{21}^f X_p + R_{22}^f Y_p + R_{23}^f Z_p + T_y}{R_{31}^f X_p + R_{32}^f Y_p + R_{33}^f Z_p + T_z} \right)$$

and C is the error covariance matrix. We will continue with the assumption that $C = I$.

Call the objective function $\Phi(u) = \epsilon(u)^{\mathrm{T}} \epsilon(u)$ with u the vector of unknowns. Given a starting value for the vector of unknowns u we iterate with steps Δu by locally fitting a quadratic function to $\Phi(u)$

$$\Phi(u + \Delta u) = \Phi(u) + \Delta u^{\mathrm{T}} \nabla \Phi(u) + \frac{1}{2} \Delta u^{\mathrm{T}} \mathbf{H}(u) \Delta u \,,$$

where $\nabla \Phi$ is the gradient and $\mathbf{H}$ is the Hessian of Φ. The minimum of this local quadratic is at Δu satisfying

$$\mathbf{H} \delta u = -\nabla \Phi(u) \,.$$

If $\Phi(u) = \epsilon(u)^{\mathrm{T}} \epsilon(u)$ then

$$\nabla \Phi = 2 \sum_i \epsilon_i(u) \nabla \epsilon_i(u)^{\mathrm{T}} = \mathbf{J}(u)^{\mathrm{T}} \epsilon \,,$$

where the Jacobian $\mathbf{J}$ consists of elements

$$\mathbf{J}_{ij} = \frac{\partial \epsilon_i}{\partial u_j} \,,$$

and the Hessian reads

$$\mathbf{H} = 2 \sum_i \left(\nabla \epsilon_i(u) \nabla \epsilon_i(u)^{\mathrm{T}} + \epsilon_i(u) \frac{\partial^2 \epsilon_i}{\partial u^2} \right)$$
$$= 2 \left(\mathbf{J}(u)^{\mathrm{T}} \mathbf{J}(u) + \sum_i \epsilon_i(u) \frac{\partial^2 \epsilon_i}{\partial u^2} \right) \approx 2 \mathbf{J}(u)^{\mathrm{T}} \mathbf{J}(u)$$

by omitting quadratic terms inside the Hessian. This yields the Gauss–Newton iteration

$$(\mathbf{J}^{\mathrm{T}} \mathbf{J}) \Delta u = \mathbf{J}^{\mathrm{T}} \epsilon \,,$$

involving the inversion of a $(6F + 3N - 7) \times (6F + 3N - 7)$ matrix. Bundle adjustment is about the *art* of inverting efficiently $(\mathbf{J}^{\mathrm{T}} \mathbf{J})$.

Let us split the unknown vector u into $u = (a, b)$ following [32.48] obtaining

- $6F - 6$ motion unknowns a,
- $3P - 1$ structure unknonws b,

and we will explain this case better if we assume two motion unknowns a_1 and a_2 corresponding to 2 frames, and 3 unknown points b_1, b_2, b_3.

For keeping symmetry in writing we do not deal here with the global reference and the global scale ambiguity.

The Jacobian for 2 frames and 3 points has 6 pairs of rows (one pair for each image projection) and 15 columns/unknowns

$$\mathbf{J} = \frac{\partial \epsilon}{\partial (a, b)} = \begin{pmatrix} \mathbf{A}_1^1 & 0 & \mathbf{B}_1^1 & 0 & 0 \\ 0 & \mathbf{A}_1^2 & \mathbf{B}_1^2 & 0 & 0 \\ \mathbf{A}_2^1 & 0 & 0 & \mathbf{B}_2^1 & 0 \\ 0 & \mathbf{A}_2^2 & 0 & \mathbf{B}_2^2 & 0 \\ \mathbf{A}_3^1 & 0 & 0 & 0 & \mathbf{B}_3^1 \\ \underbrace{0}_{\text{motion}} & \underbrace{\mathbf{A}_3^2}_{\text{motion}} & \underbrace{0}_{\text{structure}} & \underbrace{0}_{\text{structure}} & \underbrace{\mathbf{B}_3^2}_{\text{structure}} \end{pmatrix},$$

with $\mathbf{A}$ matrices being 2×6 and $\mathbf{B}$ matrices being 2×3 being Jacobians of the error ϵ_i^f of the projection of the i-th point in the f-th frame. We observe now a pattern emerging in $\mathbf{J}^{\mathrm{T}} \mathbf{J}$

$$\mathbf{J}^{\mathrm{T}} \mathbf{J} = \begin{pmatrix} \mathbf{U}^1 & 0 & \mathbf{W}_1^1 & \mathbf{W}_2^1 & \mathbf{W}_3^1 \\ 0 & \mathbf{U}^2 & \mathbf{W}_1^2 & \mathbf{W}_2^2 & \mathbf{W}_3^3 \\ .. & .. & \mathbf{V}_1 & 0 & 0 \\ .. & .. & 0 & \mathbf{V}_2 & 0 \\ .. & .. & 0 & 0 & \mathbf{V}_3 \end{pmatrix},$$

with the block diagonals for motion and structure separated. Let us rewrite the basic iteration $(\mathbf{J}^{\mathrm{T}} \mathbf{J}) \Delta u = \mathbf{J}^{\mathrm{T}} \epsilon$ as

$$\begin{pmatrix} \mathbf{U} & \mathbf{W} \\ \mathbf{W}^{\mathrm{T}} & \mathbf{V} \end{pmatrix} \begin{pmatrix} \Delta a \\ \Delta b \end{pmatrix} = \begin{pmatrix} \epsilon_a' \\ \epsilon_b' \end{pmatrix},$$

and premultiply with

$$\begin{pmatrix} I & \mathbf{W}\mathbf{V}^{-1} \\ 0 & I \end{pmatrix} \begin{pmatrix} \mathbf{U} & \mathbf{W} \\ \mathbf{W}^{\mathrm{T}} & \mathbf{V} \end{pmatrix} \begin{pmatrix} \Delta a \\ \Delta b \end{pmatrix}$$
$$= \begin{pmatrix} I & \mathbf{W}\mathbf{V}^{-1} \\ 0 & I \end{pmatrix} \begin{pmatrix} \epsilon_a' \\ \epsilon_b' \end{pmatrix}$$

We find out that motion parameters can be updated separately by inverting a $6F \times 6F$ matrix

$$(\mathbf{U} - \mathbf{W}\mathbf{V}^{-1}\mathbf{W}^{\mathrm{T}}) \Delta a = \epsilon_a' - \mathbf{W}\mathbf{V}^{-1} \epsilon_b' \,.$$

Each 3-D point can be updated separately by inverting a 3×3 matrix $\mathbf{V}$

$$\mathbf{V} \Delta b = \epsilon_b' - \mathbf{W}^{\mathrm{T}} \Delta a$$

Avidbots Ex1016
Page 853 of 2154

It is worth mentioning that bundle adjustment though extremely slow captures the correlation between motion estimates and structure (3-D points) estimates which is artificially hidden in the iterative scheme in (32.29).

The largest scale motion estimation and registration of views has been performed by *Teller* et al. [32.49] with a decoupled computation first of relative rotations and finally of relative translations. The above multiple view SfM techniques can also be applied in a sliding window mode in time. *Davison* et al. [32.28] showed the first real-time recursive approach by decoupling the direction of the viewing rays from the depth unknowns. For other recursive approaches the reader is referred to the corresponding SLAM chapter.

32.2 3-D Vision for Grasping

In this section we will move from the basic geometry required for grasping to the main 3-D vision challenges associated with the limited knowledge we might have about the shape of the object as well as the actual selection of 3-D grasping poses.

Naturally, object grasping and manipulation is closely related to general scene understanding and problems such as object detection, recognition, categorization and pose estimation. Taking all the above, there are very few approaches that address all the problems in a single system. One example, reported in [32.50], addresses the problem of enabling transfer of grasp knowledge between object categories, defined using both their physical properties and functionality. This is a challenging problem given that a number of objects with similar physical properties afford different tasks. An example can be a screwdriver and a carrot that are structurally alike, but only the former can be used as a tool, or a ball and an orange where only the latter affords eating (Fig. 32.3).

In relation to object grasping in particular, there are methods that assume that full 3-D model of the object is available and concentrate on grasp synthesis solely. In addition, many of the approaches conduct experiments in a simulated environment without working with real sensory data. However, the knowledge generated in simulation can also be applied later onto sensory data. Another group of approaches considers grasp synthesis on real sensory data directly, dealing with problems such as noise, occlusions and missing data.

If the object to be grasped is known, there are approaches that store a database of grasp hypotheses, generated either in simulation or through experiments in a real setting. Most of the approaches assume that a 3-D mesh of the object is available and the challenge is then to automatically generate a set of feasible grasp hypotheses. This involves sampling the infinite space of possible hand configurations and ranking the resulting grasps according to some quality metric.

To simplify the process, a common approach is to approximate object's shape with a constellation of primitives such as spheres, cones, cylinders, boxes or superquadrics [32.51–55]. The purpose of using shape primitives is to reduce the number of candidate grasps and thus prune the search space for finding the optimal set of grasp hypotheses.

One example, shown in Fig. 32.4 and reported in [32.52], decomposes a point cloud from a stereo camera into a constellation of boxes. Grasp planning is performed directly on the boxes which reduces the number of potential grasps. *El-Khoury* and *Sahbani* [32.56] distinguish between graspable and nongraspable parts of an object where each part is represented by fitting a superquadric to the point cloud data. *Pelossof* et al. [32.57] approximate an object with a single superquadric and use a Support Vector Machines based approach to search for the grasp that maximizes the grasp quality. *Boularias* et al. [32.58] model an object as a Markov random field (MRF) in which the nodes are points from the point cloud and edges are spanned between the six nearest neighbors of a point. A node in the MRF carries either one of the two labels: a good or a bad grasp location. *Detry* et al.[32.59] model the object as a constellation of local multimodal contour descriptors. The set of associated grasp hypotheses is modeled as a nonparametric density function in the space of six-dimensional (6-D) gripper poses, referred to as a *bootstrap* density. *Papazov* et al. [32.60] demon-

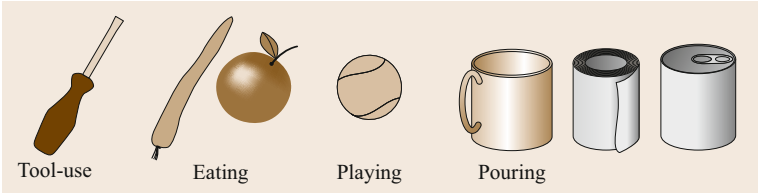


Fig. 32.3 Examples of physically similar objects that afford different tasks

Avidbots Ex1016
Page 854 of 2154

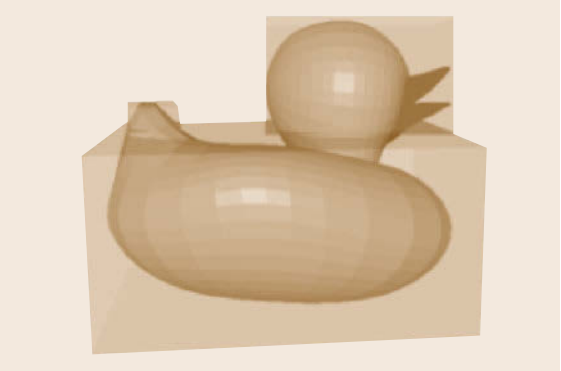


Fig. 32.4 Generation of grasp candidates through object shape approximation and decomposition from (after [32.52])

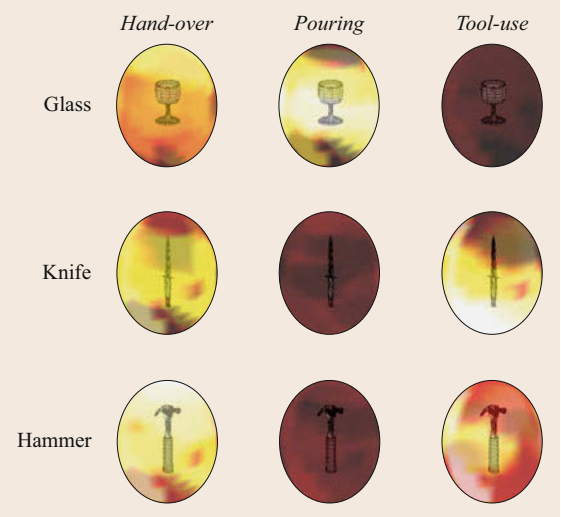


Fig. 32.5 Ranking of approach vectors on different objects given a specific task. The brighter an area the higher the rank. The darker an area, the lower the rank (after [32.62])

strates 3-D object recognition and pose estimation in a grasping scenario considering cluttered scenes. *Weisz* and *Allen* [32.61] proposes a metric suitable for predicting grasp stability under pose uncertainty.

There are several approaches that deal specifically with incomplete point clouds. *Marton* et al. [32.63] exploit symmetry by fitting a curve to a cross section of the point cloud. *Rao* et al. [32.64] concentrates of depth segmentation and sample grasp points from the surface of a segmented object using surface normals. *Bohg* et al. [32.65] presents a related approach that reconstructs full object shape assuming planar symmetry and generates grasps based on the global shape of the object. *Bone* et al. [32.66] makes no prior assumption about the shape of the object and apply shape carving for generating a parallel-jaw gripper grasps. *Hsiao* et al. [32.67] employs heuristics for generating grasp hypotheses dependent on the shape of the point cloud. Recent work in [32.68] identifies regions that afford force closure grasps by evaluating local curvature of the objects to create an initial opposing grasp with two or three fingers, dependent on the relative size of the object with respect to the hand. *Richtsfeld* and *Vincze* [32.69] uses a stereo-camera setup to generate a 3-D representation of a scene with several objects and then generates various top grasps on object candidates. *Maldonado* et al. [32.70] use time-of-flight range data, model objects using 3-D Gaussians and rely on finger torque information during grasping to monitor the grasp execution. *Stückler* et al. [32.71] generate grasp hypotheses based on eigenvectors of the object's *footprints* that are generated by projecting the 3-D object point cloud onto the supporting surface. The work of [32.72] presents a system for general scene understanding used for grasp planning and execution. The system uses a bottom-up grouping approach where contour and surface structures are used as the basis for grasp planning. The work builds upon previous work presented in [32.73].

Most of the recent work concentrates on grasp generalization either by observing human grasping or through off- and on-line learning directly on the robot. *Kroemer* et al. [32.74] demonstrates generalization capabilities using a pouring task scenario. The goal of the approach is to find a part of the object that is most likely to afford the demonstrated action. The learning method is based on the kernel logistic regression. *Herzog* et al. [32.75] stores a set of local templates of object that a human is interacting with. If a local part of an object segmented online is similar to a template in the database, the associated grasp hypothesis is executed. *Song* et al. [32.62] approach the problem of inferring a full grasp configuration in relation to a specific task the object is intended for. As in [32.76], the joint distribution over various grasping variables is modeled as a Bayesian network. Additional variables like task, object category and task constraints are introduced. The structure of this model is learned given a large number of grasp examples generated in a simulator and annotated with grasp quality metrics as well as suitability for a specific task. The learned quality of grasps on specific objects given a task is visualized in Fig. 32.5.

Avidbots Ex1016
Page 855 of 2154

32.3 Conclusion and Further Reading

As main additional sources of reading, we recommend the textbooks by *Hartley* and *Zisserman* [32.12], *Ma* et al. [32.46], *Faugeras* [32.77], and *Faugeras* and *Luong* [32.11]. The reader is referred to Chap. 5 for fundamentals of estimation, to Chap. 35 for sensor fusion, to Chap. 34 for visual servoing, to Chap. 31 for range sensing, to Chap. 45 for 3-D models of the world, and to Chap. 46 for SLAM.

3-D vision is a rapidly advancing field and in this chapter we have covered only geometric approaches based on RGB cameras. Although depth sensors will become ubiquitous indoors and might be outdoors as well, RGB cameras remain formidable because of the higher number and larger diversity of features that can be matched and used for pose estimation and 3-D-modelling. Long range sensing can still be covered from motion with large translation while active sensors are constrained in terms of energy reflected from the environment.

Video-References

VIDEO 120 Google's project Tango
available from http://handbookofrobotics.org/view-chapter/32/videodetails/120

VIDEO 121 Finding paths through the world's photos
available from http://handbookofrobotics.org/view-chapter/32/videodetails/121

VIDEO 122 LIBVISO: Visual odometry for intelligent vehicles
available from http://handbookofrobotics.org/view-chapter/32/videodetails/122

VIDEO 123 Parallel tracking and mapping for small AR workspaces (PTAM)
available from http://handbookofrobotics.org/view-chapter/32/videodetails/123

VIDEO 124 DTAM: Dense tracking and mapping in real-time
available from http://handbookofrobotics.org/view-chapter/32/videodetails/124

VIDEO 125 3-D models from 2-D video – automatically
available from http://handbookofrobotics.org/view-chapter/32/videodetails/125

References

32.1 S. Izadi, R.A. Newcombe, D. Kim, O. Hilliges, D. Molyneaux, S. Hodges, P. Kohli, J. Shotton, A.J. Davison, A. Fitzgibbon: Kinectfusion: Real-time dynamic 3D surface reconstruction and interaction, ACM SIGGRAPH 2011 Talks (2011) p. 23

32.2 Google: Atap project tango, https://www.google.com/atap/projecttango (2014)

32.3 J.A. Hesch, D.G. Kottas, S.L. Bowman, S.I. Roumeliotis: Camera-IMU-based localization: Observability analysis and consistency improvement, Int. J. Robotics Res. **33**(1), 182–201 (2014)

32.4 N. Snavely, S.M. Seitz, R. Szeliski: Modeling the world from internet photo collections, Int. J. Comput. Vis. **80**(2), 189–210 (2008)

32.5 Z. Kukelova, M. Bujnak, T. Pajdla: Polynomial eigenvalue solutions to minimal problems in computer vision, IEEE Trans. Pattern Anal. Mach. Intell. **34**(7), 1381–1393 (2012)

32.6 F. Kahl, S. Agarwal, M.K. Chandraker, D. Kriegman, S. Belongie: Practical global optimization for multiview geometry, Int. J. Comput. Vis. **79**(3), 271–284 (2008)

32.7 R.I. Hartley, F. Kahl: Global optimization through rotation space search, Int. J. Comput. Vis. **82**(1), 64–79 (2009)

32.8 Z. Zhang: A flexible new technique for camera calibration, IEEE Trans. Pattern Anal. Mach. Intell. **22**, 1330–1334 (2000)

32.9 M. Pollefeys, L. Van Gool, M. Vergauwen, F. Verbiest, K. Cornelis, J. Tops, R. Koch: Visual modeling with a hand-held camera, Int. J. Comput. Vis. **59**, 207–232 (2004)

32.10 M. Pollefeys, L. Van Gool: Stratified self-calibration with the modulus constraint, IEEE Trans. Pattern Anal. Mach. Intell. **21**, 707–724 (1999)

32.11 O. Faugeras, Q.-T. Luong, T. Papadopoulo: *The Geometry of Multiple Images: The Laws That Govern the Formation of Multiple Images of a Scene and Some of Their Applications* (MIT Press, Cambridge 2001)

32.12 R. Hartley, A. Zisserman: *Multiple View Geometry* (Cambridge Univ. Press, Cambridge 2000)

32.13 K. Ottenberg, R.M. Haralick, C.-N. Lee, M. Nolle: Review and analysis of solutions of the three-point perspective problem, Int. J. Comput. Vis. **13**, 331–356 (1994)

32.14 M.A. Fischler, R.C. Bolles: Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography, ACM Commun. **24**, 381–395 (1981)

Avidbots Ex1016
Page 856 of 2154

32.15 R. Kumar, A.R. Hanson: Robust methods for estimaging pose and a sensitivity analysis, Comput. Vis. Image Underst. **60**, 313–342 (1994)
32.16 C.-P. Lu, G. Hager, E. Mjolsness: Fast and globally convergent pose estimation from video images, IEEE Trans. Pattern Anal. Mach. Intell. **22**, 610–622 (2000)
32.17 L. Quan, Z. Lan: Linear n-point camera pose determination, IEEE Trans. Pattern Anal. Mach. Intell. **21**, 774–780 (1999)
32.18 A. Ansar, K. Daniilidis: Linear pose estimation from points and lines, IEEE Trans. Pattern Anal. Mach. Intell. **25**, 578–589 (2003)
32.19 V. Lepetit, F. Moreno-Noguer, P. Fua: EPNP: An accurate $o(n)$ solution to the PNP problem, Int. J. Comput. Vis. **81**(2), 155–166 (2009)
32.20 G.H. Golub, C.F. van Loan: *Matrix Computations* (Johns Hopkins Univ. Press, Baltimore 1983)
32.21 J.A. Hesch, S.I. Roumeliotis: A direct least-squares (dls) method for pnp, IEEE Int. Conf. Comput. Vis. (ICCV) (2011) pp. 383–390
32.22 C.J. Taylor, D.J. Kriegman: *Minimization on the Lie Group SO(3) and Related Manifolds* (Yale University, New Haven 1994)
32.23 P.-A. Absil, R. Mahony, R. Sepulchre: *Optimization Algorithms on Matrix Manifolds* (Princeton Univ. Press, Princeton 2009)
32.24 Y. Ma, J. Košecká, S. Sastry: Optimization criteria and geometric algorithms for motion and structure estimation, Int. J. Comput. Vis. **44**(3), 219–249 (2001)
32.25 R.I. Hartley, P. Sturm: Triangulation, Comput. Vis. Image Underst. **68**(2), 146–157 (1997)
32.26 B. Kitt, A. Geiger, H. Lategahn: Visual odometry based on stereo image sequences with ransac-based outlier rejection scheme, IEEE Intell. Veh. Symp. (IV) (2010)
32.27 B.K.P. Horn, H.M. Hilden, S. Negahdaripour: Closed-form solution of absolute orientation using orthonormal matrices, J. Opt. Soc. Am. A **5**, 1127–1135 (1988)
32.28 A.J. Davison, I.D. Reid, N.D. Molton, O. Stasse: Monoslam: Real-time single camera SLAM, IEEE Trans. Pattern Anal. Mach. Intell. **29**(6), 1052–1067 (2007)
32.29 R. Tron, K. Daniilidis: On the quotient representation for the essential manifold, Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (2014) pp. 1574–1581
32.30 T.S. Huang, O.D. Faugeras: Some properties of the E matrix in two-view motion estimation, IEEE Trans. Pattern Anal. Mach. Intell. **11**, 1310–1312 (1989)
32.31 D. Nister: An efficient solution for the five-point relative pose problem, IEEE Trans. Pattern Anal. Mach. Intell. **26**, 756–777 (2004)
32.32 H. Li, R. Hartley: Five-point motion estimation made easy, IEEE 18th Int. Conf. Pattern Recognit. (ICPR), Vol. 1 (2006) pp. 630–633
32.33 Z. Kukelova, M. Bujnak, T. Pajdla: Polynomial eigenvalue solutions to the 5-pt and 6-pt relative pose problems, BMVC (2008) pp. 1–10
32.34 H. Stewenius, C. Engels, D. Nistér: Recent developments on direct relative orientation, ISPRS J. Photogramm. Remote Sens. **60**(4), 284–294 (2006)
32.35 D. Batra, B. Nabbe, M. Hebert: An alternative formulation for five point relative pose problem, IEEE Workshop Motion Video Comput. (2007) pp. 21–21
32.36 Center for Machine Perception, Minimal problems in computer vision; http://cmp.felk.cvut.cz/minimal/5_pt_relative.php
32.37 S. Maybank: *Theory of Reconstruction from Image Motion* (Springer, Berlin, Heidelberg 1993)
32.38 S.J. Maybank: The projective geometry of ambiguous surfaces, Phil. Trans. Royal Soc. Lond. A **332**(1623), 1–47 (1990)
32.39 A. Jepson, D.J. Heeger: A fast subspace algorithm for recovering rigid motion, Proc. IEEE Workshop Vis. Motion, Princeton (1991) pp. 124–131
32.40 C. Fermüller, Y. Aloimonos: Algorithmic independent instability of structure from motion, Proc. 5th Eur. Conf. Comput. Vision, Freiburg (1998)
32.41 K. Daniilidis, M. Spetsakis: Understanding noise sensitivity in structure from motion. In: *Visual Navigation*, ed. by Y. Aloimonos (Lawrence Erlbaum, Mahwah 1996) pp. 61–88
32.42 S. Soatto, R. Brockett: Optimal structure from motion: Local ambiguities and global estimates, IEEE Conf. Comput. Vis. Pattern Recognit., Santa Barbara (1998)
32.43 J. Oliensis: A new structure-from-motion ambiguity, IEEE Trans. Pattern Anal. Mach. Intell. **22**, 685–700 (1999)
32.44 O. Naroditsky, X.S. Zhou, J. Gallier, S. Roumeliotis, K. Daniilidis: Two efficient solutions for visual odometry using directional correspondence, IEEE Trans. Patterns Anal. Mach. Intell. (2012)
32.45 Y. Ma, K. Huang, R. Vidal, J. Kosecka, S. Sastry: Rank conditions of the multiple view matrix, Int. J. Comput. Vis. **59**(2), 115–139 (2004)
32.46 Y. Ma, S. Soatto, J. Kosecka, S. Sastry: *An Invitation to 3-D Vision: From Images to Geometric Models* (Springer, Berlin, Heidelberg 2003)
32.47 W. Triggs, P. McLauchlan, R. Hartley, A. Fitzgibbon: Bundle adjustment – A modern synthesis, Lect. Notes Comput. Sci **1883**, 298–372 (2000)
32.48 M. Lourakis, A. Argyros: *The Design and Implementation of a Generic Sparse Bundle Adjustment Software Package Based on the Levenberg–Marquard Method*, Tech. Rep, Vol. 340 (ICS/FORTH, Heraklion 2004)
32.49 S. Teller, M. Antone, Z. Bodnar, M. Bosse, S. Coorg: Calibrated, registered images of an extended urban area, Int. Conf. Comput. Vis. Pattern Recognit., Kauai, Vol. 1 (2001) pp. 813–820
32.50 D. Kragic, M. Madry, D. Song: From object categories to grasp transfer using probabilistic reasoning, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (2012) pp. 1716–1723
32.51 A.T. Miller, S. Knoop, H.I. Christensen, P.K. Allen: Automatic grasp planning using shape primitives, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (2003) pp. 1824–1829
32.52 K. Hübner, D. Kragic: Selection of robot pre-grasps using box-based shape approximation, IEEE/RSJ

Avidbots Ex1016
Page 857 of 2154

Int. Conf. Intell. Robots Syst. (IROS) (2008) pp. 1765–1770
32.53 C. Dunes, E. Marchand, C. Collowet, C. Leroux: Active rough shape estimation of unknown objects, IEEE Int. Conf. Intell. Robots Syst. (IROS) (2008) pp. 3622–3627
32.54 M. Przybylski, T. Asfour: Unions of balls for shape approximation in robot grasping, IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS), Taipei (2010) pp. 1592–1599
32.55 C. Goldfeder, P.K. Allen, C. Lackner, R. Pelossof: Grasp Planning Via Decomposition Trees, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (2007) pp. 4679–4684
32.56 S. El-Khoury, A. Sahbani: Handling objects by their handles, IEEE/RSJ Int. Conf. Intell. Robots Syst. Workshop Grasp Task Learn. Imitation (2008)
32.57 R. Pelossof, A. Miller, P. Allen, T. Jebera: An SVM learning approach to robotic grasping, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (2004) pp. 3512–3518
32.58 A. Boularias, O. Kroemer, J. Peters: Learning robot grasping from 3-d images with markov random fields, IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS) (2011) pp. 1548–1553
32.59 R. Detry, E. Başeski, N. Krüger, M. Popović, Y. Touati, O. Kroemer, J. Peters, J. Piater: Learning object-specific grasp affordance densities, IEEE Int. Conf. Dev. Learn. (2009) pp. 1–7
32.60 C. Papazov, S. Haddadin, S. Parusel, K. Krieger, D. Burschka: Rigid 3D geometry matching for grasping of known objects in cluttered scenes, Int. J. Robotics Res. **31**(4), 538–553 (2012)
32.61 J. Weisz, P.K. Allen: Pose error robust grasping from contact wrench space metrics, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (2012) pp. 557–562
32.62 D. Song, C.H. Ek, K. Hübner, D. Kragic: Multivariate discretization for bayesian network structure learning in robot grasping, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (2011) pp. 1944–1950
32.63 Z.C. Marton, D. Pangercic, N. Blodow, J. Kleinehellefort, M. Beetz: General 3D modelling of novel objects from a single view, IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS) (2010) pp. 3700–3705
32.64 D. Rao, V. Le Quoc, T. Phoka, M. Quigley, A. Sudsang, A.Y. Ng: Grasping novel objects with depth segmentation, IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS), Taipei (2010) pp. 2578–2585
32.65 J. Bohg, M. Johnson-Roberson, B. León, J. Felip, X. Gratal, N. Bergström, D. Kragic, A. Morales: Mind the gap – Robotic grasping under incomplete observation, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (2011)
32.66 G.M. Bone, A. Lambert, M. Edwards: Automated Modelling and Robotic Grasping of Unknown Three-Dimensional Objects, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (2008) pp. 292–298
32.67 K. Hsiao, S. Chitta, M. Ciocarlie, E.G. Jones: Contact-reactive grasping of objects with partial shape information, IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS) (2010) pp. 1228–1235
32.68 M.A. Roa, M.J. Argus, D. Leidner, C. Borst, G. Hirzinger: Power grasp planning for anthropomorphic robot hands, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (2012)
32.69 M. Richtsfeld, M. Vincze: Grasping of Unknown Objects from a Table Top, ECCV Workshop Vis. Action: Effic. Strateg. Cogn. Agents Complex Environ. (2008)
32.70 A. Maldonado, U. Klank, M. Beetz: Robotic grasping of unmodeled objects using time-of-flight range data and finger torque information, IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS) (2010) pp. 2586–2591
32.71 J. Stückler, R. Steffens, D. Holz, S. Behnke: Real-time 3d perception and efficient grasp planning for everyday manipulation tasks, Eur. Conf. Mob. Robots (ECMR) (2011)
32.72 G. Kootstra, M. Popovic, J.A. Jørgensen, K. Kuklinski, K. Miatliuk, D. Kragic, N. Kruger: Enabling grasping of unknown objects through a synergistic use of edge and surface information, Int. J. Robotics Res. **31**(10), 1190–1213 (2012)
32.73 D. Kraft, N. Pugeault, E. Baseski, M. Popovic, D. Kragic, S. Kalkan, F. Wörgötter, N. Krueger: Birth of the object: Detection of objectness and extraction of object shape through object action complexes, Int. J. Humanoid Robotics **pp**, 247–265 (2009)
32.74 O. Kroemer, E. Ugur, E. Oztop, J. Peters: A Kernel-based Approach to Direct Action Perception, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (2012)
32.75 A. Herzog, P. Pastor, M. Kalakrishnan, L. Righetti, T. Asfour, S. Schaal: Template-based learning of grasp selection, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (2012)
32.76 L. Montesano, M. Lopes, A. Bernardino, J. Santos-Victor: Learning object affordances: From sensory–motor coordination to imitation, IEEE Trans. Robotics **24**(1), 15–26 (2008)
32.77 O. Faugeras: *Three-Dimensional Computer Vision* (MIT Press, Cambridge 1993)

Avidbots Ex1016
Page 858 of 2154

Multimedia Contents

33. Visual Object Class Recognition

Michael Stark, Bernt Schiele, Aleš Leonardis

Object class recognition is among the most fundamental problems in computer vision and thus has been researched intensively over the years. This chapter is mostly concerned with the recognition and detection of basic level object classes such as cars, persons, chairs, or dogs. We will review the state of the art and in particular discuss the most promising methods available today.

33.1 **Object Classes** 825
33.2 **Review of the State of the Art** 826
33.2.1 Bag-of-Words Models 826
33.2.2 Spatial Pyramid Matching 828
33.2.3 Flexible Part Models 829
33.2.4 Histogram of Oriented Gradients 831
33.2.5 The Deformable Part Model 832
33.2.6 Convolutional Neural Networks 834
33.2.7 Fine-Grained Categorization 836
33.2.8 Datasets 836
33.3 **Discussion and Conclusions** 837
33.3.1 Task 837
33.3.2 Data 838
33.3.3 Efficiency 838
References 838

33.1 Object Classes

Generic object class recognition has been one of the goals of computer vision since its beginnings [33.1, 2]. In this context, it is important to emphasize that the notion and the abstraction level of object classes is far from being uniquely and clearly defined. Notably, the question of how humans organize knowledge at different levels has received much attention in cognitive psychology [33.3, 4]. Taking an example from Brown's work, a dog can not only be thought of as a *dog*, but also as a *boxer*, a *quadruped*, or in general an *animate being* [33.3]. Yet, *dog* is the term that comes to mind most easily, which is by no means accidental. Experiments show that there is a *basic level* in human categorization at which most knowledge is organized [33.5]. According to *Rosch* et al. and *Lakoff* [33.5, 6], this basic level is also:

- The highest level at which category members have similar perceived shape
- The highest level at which a single mental image can reflect the entire category
- The highest level at which a person uses similar motor actions for interacting with category members
- The level at which human subjects are fastest at identifying category members
- the first level named and understood by children.

Given that the basic level of categorization is easiest for humans, it should be a good starting point for machine vision, too [33.7]. The next lower level, *subordinate categories*, corresponds to the individual level used in object identification, e.g., the next higher level, *super-ordinate categories*, requires a higher degree of abstraction and world knowledge. The following will thus discuss state-of-the-art approaches first for basic-level categorization but then also briefly touch upon subordinate categorization for fine-grained classification.

Avidbots Ex1016
Page 859 of 2154

33.2 Review of the State of the Art

One can think of all current methods in object class recognition as being part-based methods, in which object class instances are described by a set of parts (such as human body parts, for instance), and the topology between the parts (such as the kinematic tree defined by the joints that connect the human body parts). The following sections give a systematic treatment of different instantiations of state of the art, part-based methods, ranging from the bag-of-words model (Sect. 33.2.1) and spatial pyramid matching (Sect. 33.2.2), over flexible part models (Sect. 33.2.3) and rigid histogram of oriented gradients templates (Sect. 33.2.4) to deformable part models (Sect. 33.2.5) and convolutional neural networks (Sect. 33.2.6). Table 33.1 summarizes the various part-based methods presented in this section. Later sections highlight recent trends in fine-grained categorization (Sect. 33.2.7) and datasets for visual recognition (Sect. 33.2.8).

Table 33.1 Classification of some prominent approaches as part-based models

Model	Parts	Topology
Bag-of-words	Feature clusters	No spatial information coded
Spatial pyramid	Feature clusters	Hierarchical but rigid spatial grid
Constellation model	Feature clusters	Full covariance matrix between clusters
Implicit shape model	Feature clusters	Star-shaped model
HOG	HOG-cells	Rigid spatial grid
DPM	HOG-parts	Star-model
Pictorial structures model	Semantic parts	Tree structure
Convolutional neural networks	Convolution filters	Spatial spooling and hierarchy

33.2.1 Bag-of-Words Models

Apart from allowing to establish pointwise correspondences between multiple images of the same scene, local image patches also provide the foundation for a variety of object class representations for visual recognition. Inspired by the natural language processing literature, the *bag-of-words* (BOw) or *bag of features* model [33.8] represents an image as an unordered collection (a bag) of local patches. This representation assumes that most of the important information about the image content is captured purely by which patches occur (and how often), irrespective of where they occur. While this is clearly an approximation, it seems reasonable from the text document analogy: even if the words in a text document are shuffled randomly, the mere occurrence of certain key terms provides a strong cue about the document's topic. Likewise, a collection of local image patches, such as two eye patches, mouth, and nose, provide strong cues for the presence of a face in an image.

The bag-of-words model is typically based on a codebook of visual words: a fixed, discrete set of visual features relative to which all others are described (Fig. 33.1c). An input image (Fig. 33.1a) can then be described relative to that codebook, by counting how often each codeword appears (Fig 33.1b). In the following paragraphs, we describe in detail the extraction of visual features, the generation of codebooks of visual words, and their use in image classification using probabilistic generative and discriminative models.

Feature Extraction

Locations and scales for feature extraction are typically either picked sparsely, based on an interest operator, such as the difference of Gaussian (DOG)

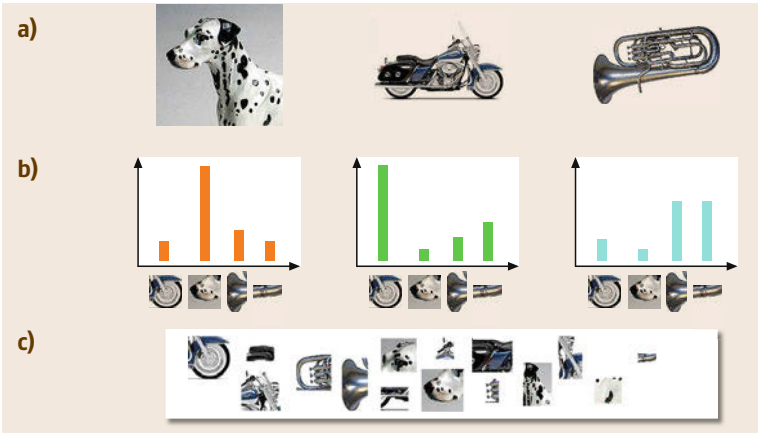


Fig.33.1a–c The bag-of-words representation. **(a)** Image; **(b)** BOw histogram; **(c)** codebook of words

Avidbots Ex1016
Page 860 of 2154

interest point detector for scale invariant feature transform (SIFT) [33.9], Harris corners, scale- and affine-invariant Hessian–Laplace interest points [33.10], maximally stable extremal regions (MSER) [33.11], or based on dense feature sampling using a fixed spatial grid [33.12]. While interest operators generally provide well-localized, repeatable features with sufficient recall, the increasing availability of compute power today has made very dense feature sampling feasible (e.g., with a stride of only one pixel, resulting in many thousands of features in an image) – in the limit, dense sampling typically outperforms sparse interest operators in image classification as interest point detectors are designed to sample particular types of image structure whereas dense sampling allows to sample any type of image structure (note that this might be different for other applications, such as structure from motion, in which it is more important to find smaller numbers of features with high precision).

Codebook Generation

Codebooks of visual features are typically constructed by applying an unsupervised clustering algorithm, such as *k*-means or hierarchical agglomerative clustering [33.13], to a training set of image feature vectors. The result is a vector-quantized representation of the visual feature space that groups similar feature descriptors into clusters. Each cluster thus effectively generalizes over its members, such that small visual differences are suppressed – this is often seen as a crucial trait for categorization even though larger codebook sizes typically result in better performance. Each cluster is identified with a representative member, i. e., its centroid (for *k*-means) or medoid (for agglomerative clustering). In order to represent a new image with respect to the generated codebook, image features are extracted, and matched to the representative of each cluster (e.g., by computing pairwise distances and defining a matching threshold). The result of this matching is then recorded (e.g., as a histogram over the number of times each cluster has been matched) to yield the image or object representation. Note that in the context of BOw models the terms object and image representation are synonymous.

For codebook generation, the number of clusters is an important free parameter: it has been found that large codebook sizes typically improve recognition performance, since they represent the original feature space more faithfully, with lower reconstruction error. On the other hand, large codebooks generalize less, and come at the cost of increased computational cost when building bag-of-words representations and classifying them with nonlinear classifiers. Typically, codebooks with several and even 10 000s (multiples of ten thousand) of clusters are chosen for optimal performance [33.14].

Generative Models

Inspired by their success in the text domain, generative probabilistic models have also been adopted for image classification in connection with BOw representations. The arguably simplest one is the naive Bayes model [33.8, 15], in which the posterior probability of an image having a certain class factors into independent contributions from each histogram bin (i. e., for a given class, knowing about the presence of one feature does not make the presence of another feature any more or less likely). While this model is agnostic of correlations that often exist between features (for instance, having already observed two wheel features in a sideview of a car makes it unlikely to observe one more, assuming that only a single car is present in the image), it often works well in practice and is appealing due to its simplicity.

In line with the text domain, more powerful models have been proposed for image classification that explicitly capture the co-occurrence of features through a generative process. Here, the features contained in an image are assumed to be generated in a sequence of steps, where each step determines the choices available for the next step. More precisely, each step is implemented by sampling from a probability distribution, conditioned on the outcome of the previous step.

A popular class of generative models is the *topic model* [33.16, 17]: an image is considered to be an orderless document of visual words, composed of multiple topics (in an image depicting a beach scene, there could be a water topic, a sand topic, and a sky topic, for instance). In order to generate a particular feature in the image, a topic is first sampled from a discrete distribution over topics (a beach scene could be a roughly uniform distribution over water, sand, and sky). Then, a visual feature is sampled from a discrete distribution over features, conditioned on the previously sampled topic (e.g., for a water topic, the sampled feature is likely to be a blue image patch with a ripple pattern). In practice, the assignment of visual features to topics is represented through latent variables, which means that it has to be inferred from other quantities that can be observed (such that the likelihood of the features is maximized). This is typically implemented by means of the expectation maximization (EM) [33.18] algorithm, which alternates (re-)estimation of topic distributions (E-step) and imputation of latent variables (M-step) in an iterative procedure.

In the literature, there exists a large variety of topic models, ranging from the basic probabilistic latent semantic analysis (PLSA) [33.16] over its Bayesian extension, latent Dirichlet allocation (LDA) [33.17], to

Avidbots Ex1016
Page 861 of 2154

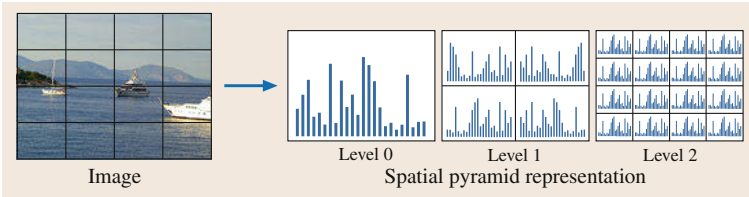


Fig. 33.2 Spatial pyramid matching as a multilevel BOw representation (after [33.20])

infinite hierarchical models of images and their features [33.19].

Discriminative Models

Once an image is represented as a histogram of matched features with respect to a codebook, it can be classified by means of any discriminative classifier that is applicable to histograms. Popular choices include support vector machines [33.21] with histogram intersection kernels or their approximate counterpart [33.22], boosting [33.23], or random forest classifiers [33.24].

Extensions

While the bag-of-words representation inherently ignores spatial information, it can be re-introduced by restricting the support of a bag-of-words representation to a local image region, and processing multiple regions independently. Specifically, instances of an object class can be localized by sliding a rectangular window over the image under consideration, extracting a bag-of-words representation at each location, and feeding the result into a classifier that distinguishes the object class of interest from a background. Objects of different sizes can be localized by applying this scheme to multiple re-scaled versions of the input image [33.25] (called an image pyramid).

Rather than applying the sliding window in a brute force fashion, it has been realized that the classifier score that an image window can attain can be bounded from above by the scores of the subwindows that it contains. This observation gives rise to an efficient subwindow search [33.26] that greatly reduces the number of classifier evaluations needed in connection with sliding window bag of words representations, making powerful but slow nonlinear kernel SVMs applicable.

33.2.2 Spatial Pyramid Matching

Instead of completely ignoring the spatial layout of local features in an image, it seems reasonable to include at least a rough notion of locality in order to increase the discriminative power of an object class representation. Spatial pyramid matching implements this notion as an extension of the bag-of-words model (Sect. 33.2.1), by extracting multiple bag-of-words representations for a single input image at different spatial locations and scales, and aggregating these local representations into a single global representation for the entire image. Specifically, spatial pyramid matching (SPM) [33.27] extracts bag-of-words histograms for each cell in a rectangular grid overlaid onto the input image (typically, coarse grids of 2^l for $l \in \{0, 1, 2\}$). This process is repeated multiple times with different numbers of cells (e.g., four by four plus two by two plus a single cell), forming an overcomplete pyramid structure of decreasing spatial resolution (Fig. 33.2). The resulting bag-of-words histograms for each cell are concatenated (and re-normalized). The appeal of SPM lies in representing the spatial layout at multiple levels of resolution without committing to any one in particular – instead, the representations of all resolutions are kept, and have the chance to contribute to image classification in case they prove useful.

The basic SPM methodology appears in numerous variations in the literature, but often follows the same pipeline of four steps (feature extraction, feature coding, spatial pooling, and classification), each of which we review in the following sections.

Feature Extraction

Like the basic bag-of-words model, the SPM model has been used in connection with various feature types, ranging from SIFT [33.9] appearance features to local edge configurations [33.28] and self-similarity features [33.29].

Feature Coding

Representing visual features relative to a pretrained codebook obtained by vector quantization is only one special case of feature coding (performing a change of representation), and several improvements have been proposed in the literature [33.30–32]. Most importantly, it has proven beneficial to explicitly formulate the codebook training and feature coding as a joint optimization problem, such that the original features can be reconstructed with small error from their respective codes. In addition, this formulation allows us to relax the assumption that each feature is matched to exactly one codebook entry by replacing the cardinality constraint with an L1 regularization. The resulting sparse coding (SC) formulation [33.31] has been shown to outperform the original SPM [33.27] by significant margins, even

Avidbots Ex1016
Page 862 of 2154

when combined with a linear SVM (while SPM works best in connection with nonlinear histogram intersection or chi-square kernels, which are computationally expensive). An improved version of SC is given by locality constrained linear coding (LLC) [33.32]. It replaces sparsity by locality constraints that encourage features to be reconstructed by nearby features in feature space, such that similar features yield similar codes (which is obviously desirable in order to facilitate classifier learning).

Instead of relying solely on counting the number of occurrences of visual features in a spatial region as promoted by SPM, SC, and LLC, the Fisher vector [33.30] method characterizes that region by how well a pretrained generative probabilistic model (a Gaussian mixture model, GMM) explains the contained features. Specifically, it builds a vector from the derivatives of the log-likelihood of the image features, given the mixture coefficients, means, and diagonal covariances of a GMM. Note that this vector has much more entries than a histogram of occurrences using the same GMM as a codebook, allowing to build highly discriminative representations already for small numbers of GMM mixture components (typically only a few hundred).

Both sparse coding and Fisher vector representations haven proven to be very effective for image classification in connection with spatial pyramids, and have been successfully applied also in large-scale settings, such as classification on ImageNet [33.33] (Sect. 33.2.8).

Spatial Pooling

Apart from the classic pyramid scheme [33.27] in which histograms of different pyramid levels are concatenated, retaining the maximum value of a particular bin over a spatial region or even multiple pyramid levels (max-pooling) has gained increasing attention [33.31], in particular in connection with CNNs (Sect. 33.2.6). Max-pooling has the advantage that it promotes invariance to local translations and scale changes, and has been found to be present in the human visual cortex (V1) [33.34].

In addition to pooling features in the two-dimensional (2-D) image plane, more object-centric pooling techniques have been developed in the context of fine-grained categorization [33.35, 36] (Sect. 33.2.7).

Classification

The last stage in an SPM pipeline is typically given by a discriminative classifier that is trained on a training set of fixed length vectors created from the SPM representations. Depending on the sophistication of the feature coding step, support vector machines with nonlinear histogram intersection or chi-square [33.27] or linear kernels [33.30–32] achieve state-of-the-art performance. Further extensions include the use of multiple kernels with learned contribution weights [33.37]. When evaluated as part of a classification cascade, multiple kernel learning (MKL) can also be used in a sliding window object class detector [33.38].

33.2.3 Flexible Part Models

While orderless bags of visual features (Sect. 33.2.1) or spatial pyramids with a fixed grid layout (Sect. 33.2.2) have proven to be effective object class representations for classification, they both fail to explicitly capture the flexible nature of physical objects that are composed of multiple parts – as an example, consider the human face with eyes, eyebrows, mouth, nose, and ears: obviously, the precise shape and spatial arrangement of these parts will vary from person to person, but variations in shape and arrangement are constrained to lie within a plausible range (i. e., the parts are not unordered as in a bag-of-words representation). It also seems intuitive that the constraints that govern the range of plausible variations should be soft, and not bound to a fixed binning or a fixed spatial grid (as in spatial pyramid matching). In this section, we review a family of flexible part models that implement this intuition.

Constellation Model

The constellation model [33.39–41] represents an object class as a flexible arrangement of a fixed number of parts. It consists of two components: first, a model of the appearance of each part, and second, a model of the relative spatial layout of the parts. It is formulated in a fully probabilistic way, such that each part is characterized by a probabilistic, generative model of its appearance, and the spatial layout of the parts is governed by a joint Gaussian distribution that captures how the location (and scale) of each part is expected to vary with the location of each other part. Interestingly, the definition of what constitutes a part is driven entirely by the training data: parts are chosen and placed such that the likelihood of the training data, given the model, is maximized during training. This can be seen as an instance of a missing data problem (the association between features and parts is unknown), and solved by means of the expectation maximization (EM) algorithm.

Unfortunately, the representative power of the joint Gaussian spatial distribution comes at a cost: the fully connected dependency structure demands examining each of the exponentially many combinations of placing all parts in an image in order to find the global optimum – the exact inference problem is known to be intractable. As a consequence, the original constellation

Avidbots Ex1016
Page 863 of 2154

model [33.40, 41] is limited to a few dozen candidates per part, per image, limiting its generality. An alternative approach to solving the exact inference problem is to resort to an approximation, e.g., by sampling likely constellations from part detection proposals [33.42]. In this way, the number of part candidates can be scaled up to several thousands.

Due to its probabilistic nature, the constellation model has also been extended to a full Bayesian model [33.43], in which the object class model itself is governed by a prior distribution over models. This abstraction allows to separate characteristics of one particular object class (e.g., the particular shape of a horse's head) from properties that are shared by classes (e.g., all quadrupeds are four legged). These shared properties can be re-used in multiple object class detectors (e.g., for horses, cows, dogs, etc.), facilitating learning from limited training data. The underlying idea of re-use of information is known as transfer learning in the machine learning literature, and has been successfully applied for image classification [33.43] as well as object class detection.

Implicit Shape Model

The implicit shape model (ISM) [33.44] relaxes the assumption of the constellation model that the locations and scales of all parts are mutually dependent. Instead, it limits the dependency structure to a star topology, in which part locations and scales are conditionally independent, given the object center (i. e., given the object center, revealing information about the location of one part does not alter the expectation of where to observe any other part). While this model clearly constitutes an approximation of the physical world, it greatly simplifies the inference problem of finding the configuration with maximum score (there exists an algorithm that is exact and efficient).

Specifically, the ISM represents an object class as a collection of local feature matches – the supply of matchable features is given by a codebook of visual words, as it is also used in bag-of-words (Sect. 33.2.1) or spatial pyramid representations (Sect. 33.2.2). In contrast to those, however, each ISM codebook entry also maintains a list of feature matches from a training set of images: each match specifies where the match occurred relative to the object center. As a result, this list of occurrences constitutes a nonparametric probability density estimate of the location of the object center, given the matched feature – in other words, each feature can cast votes for likely object center positions (e.g., a wheel feature can cast two votes: one to the left of it, assuming that it is the front wheel of a car, and one to the right, assuming it's the back wheel). This voting procedure can directly be translated into a recognition algorithm: at recognition time, all test image features cast their votes into a three-dimensional (3-D) voting space (x, y, and scale). All votes are accumulated, and since multiple features are likely to agree on the true object center, the maxima of this voting space can be considered candidate object detections (note that this voting can be considered a generalized Hough transform). In practice, continuous voting space maxima are found using a mean-shift algorithm.

In order to increase its discriminative power, the basic ISM model is extended by a verification step [33.44], in which object detection candidates are verified based on how consistent they are with respect to foreground–background segmentation. To that end, feature occurrences stored as part of the codebook are enriched with segmentation mask fragments (which have to be provided as additional supervision during training). At recognition time, the consistency of the cast segmentation mask votes is quantified based on minimum description length (MDL). In addition, it has been suggested to learn weights that regulate the contributions of feature matches to the final detection hypotheses in a discriminative way (i. e., optimized to distinguish object and background firings). This *discriminative Hough transform* [33.45] further improves the performance of the ISM and constitutes the basis for the *poselet* framework [33.46], in which a multitude of small HOG [33.47] detectors (Sect. 33.2.4) are combined in order to detect people.

Since, in theory, the ISM voting space can accommodate arbitrary quantities in addition to object center and scale, there have been extensions to predict the walking poses of pedestrians [33.48] as well as predicting the viewpoint from which the detected object is imaged [33.49, 50].

Pictorial Structures

While being the first to conceptualize the idea of objects being composed of flexible parts, the pictorial structures model [33.39, 51] is typically understood as a simplified version of the constellation model. The full mutual dependency structure between parts is relaxed to a tree (or a mixture of trees in more advanced incarnations) – note that the star-shaped ISM (Sect. 33.2.3) is hence a special case of a pictorial structures model with tree-depth one. In contrast to the constellation model, where parts are defined in a data-driven way, the pictorial structures model is typically associated with a semantic notion of parts. An intuitive example is the set of human body parts (head, torso, upper arm, lower arm, etc.) in connection with the natural kinematic tree structure (Fig. 33.3).

Avidbots Ex1016
Page 864 of 2154

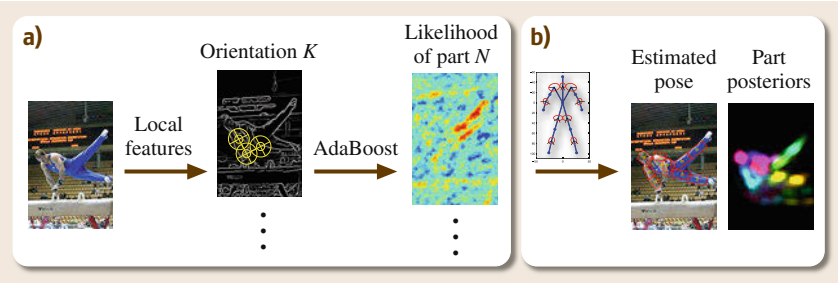


Fig.33.3a,b Human body pose estimation with the pictorial structures model. **(a)** Image likelihoods for individual parts; **(b)** final pose estimate using spatial layout prior

In fact, current state-of-the-art systems for people detection and human body pose estimation are typically built upon the pictorial structures model [33.52]. Here, pairwise relations between body parts express the angular constraints of the human body as well as the prior probability of observing people in various poses (e.g., it is unlikely to observe extreme angles at certain joints, Fig. 33.3b). In order to compensate for the weaker spatial model in comparison to the constellation model, the pictorial structures model typically relies on a strong model for the appearance of each part that is trained discriminatively (distinguishing the part from background, Fig. 33.3a) rather than in a generate way (explaining the part's appearance). Examples of discrimative part appearance models include AdaBoost [33.23] classifiers trained on dense shape context features [33.52]. Also note that part appearance and spatial layout models are trained independently (joint training has been realized to bear large potential for improvement in the context of the deformable part model [33.53] which we will review in Sect. 33.2.5).

The use of semantic parts naturally demands to provide additional supervision in the form of part annotations during training; human body parts are typically annotated with a fixed size relative to the size of the person. For recognition, inference in the tree-structured probabilistic graphical model can be carried out exactly and efficiently by max-product or sum-product belief propagation, depending on whether the maximum a posteriori (MAP) or marginal solution is preferred.

Extensions to the pictorial structures model include learning the tree structure instead of using the natural kinematic tree [33.54], introducing additional lateral edges in addition to the tree edges [33.55], and combining pictorial structures models with nonparametric representations of partial human poses (*poselets* [33.46]).

33.2.4 Histogram of Oriented Gradients

At the opposite end of the spectrum from the bag-of-words (Sect. 33.2.1) or flexible part models (Sect. 33.2.3) are approaches that represent an object class as a rigid template of local features with a fixed spatial layout. Conceptually, this corresponds to the finest level of a spatial pyramid in which each feature gets assigned its own exclusive set of histogram bins (with values equal to the feature itself). The by far most successful rigid template detector is the histogram of oriented gradients (HOG) [33.47] detector, and we will discuss its properties in detail in this section. Originally developed for people detection, the HOG feature has evolved into one of the most frequently used features in computer vision to date, and also constitutes the basis for the successful deformable part model (DPM) [33.53] described in Sect. 33.2.5.

Feature Extraction

The HOG feature can be seen as an extension of SIFT [33.9] from a single circular interest region to a rectangular detection window – at its core, it bins gradient directions into histograms, weighting their contributions by their respective magnitudes. Like SIFT, it has been carefully engineered and its parameters empirically tuned for performance (more recently, learning features end-to-end has received renewed attention, Sect. 33.2.6).

The first step for the computation of a HOG feature is to divide the detection window into a fixed grid of cells: a cell is the basic spatial unit of measurement for the HOG feature. The second step is to gamma-normalize the pixel content of each cell, in order to increase robustness to lighting changes. Third, the image is smoothed and image gradients computed using discrete derivative masks. Fourth, gradients are binned, such that each gradient casts weighted votes (by magnitude) according to nine different gradient orientation bins. Votes are interpolated bilinearly in both orientation and position. And fifth, the per-cell histograms are contrast normalized with respect to a larger block of surrounding cells in order to compensate for local variations in lighting and foreground–background contrast. Crucially, blocks are formed such that they overlap, which means that each cell contributes multiple times to the final feature descriptor, normalized in different ways, resulting in an over-complete represen-

Avidbots Ex1016
Page 865 of 2154

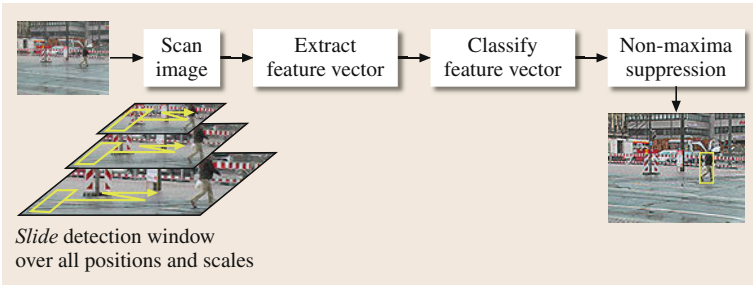


Fig. 33.4 Sliding window object class detection

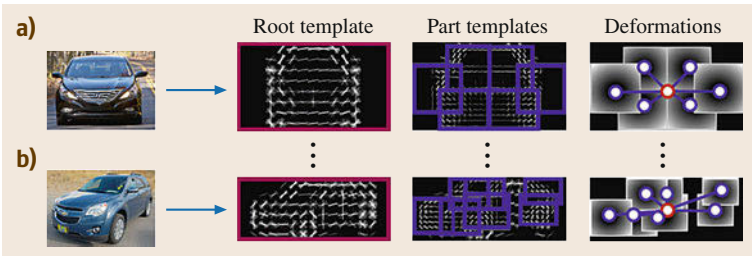


Fig.33.5a,b The deformable part model with multiple mixture components. **(a)** Mixture component for a frontal view of object class car. **(b)** Mixture component for a side view of object class car (after [33.56])

tation. This step has been found to be essential for good performance.

Sliding Window Object Detection

Similar to pure bag-of-words and spatial pyramid-based detectors, the HOG detector is typically run in a sliding window fashion over a test image, over different locations and scales (Fig. 33.4). That is, HOG features are computed for a detection window of fixed size, resulting in a feature vector that can then be fed into a discriminative classifier, such as a linear support vector machine (SVM).

Bootstrapping

One of the key ingredients for good detection performance of the HOG detector [33.47] is a technique termed as *bootstrapping*(not to be confused with a technique of the same name known from statistics, in which a dataset is repeatedly subsampled). An initially trained model is refined by re-training it on an extended set of negative training images, similar in spirit to boosting [33.23]. Specifically, the initial model is run on a validation set, and the false positive detections (wrongly detected objects) are added to the negative training set. As a result, the decision boundary of the retrained classifier is pushed further away from these *hard negatives*. Note that this procedure is not guaranteed to always lead to an improvement, but often improves performance substantially in practice.

Extensions

Due to its simple grid structure, the HOG feature lends itself to being combined with other feature channels extracted from a similar spatial grid. Notably, it has been successfully combined with optical flow features for people detection in video [33.57].

33.2.5 The Deformable Part Model

The section provides an introduction to one of the most successful computer vision algorithms to date, the DPM [33.53]. Its success is partly due to its consistently high performance for detecting a wide variety of object classes, but also due to the fact that a high quality implementation has been made publicly available from the start. The original implementation has constantly been improved and updated, and even found its way into OpenCV [33.58]. The HOG [33.47] implementation at its core has found wide spread use throughout the computer vision community.

Model Structure and Inference

The DPM can be seen as a proper generalization of the fully rigid HOG [33.47] template to a flexible, part-based object class representation (Fig. 33.5). It generalizes HOG in two different ways that both increase its ability to handle intraclass variation: first, it divides the single HOG template into a collection of multiple templates that can move relative to one another and capture small variations in object shape (hence the name *Deformable Part Model*). Specifically, it includes a low-resolution *root* template for the entire object plus a number of high-resolution *parts* that are constrained to occur not too far from their expected positions relative to the root – this constitutes a star-shaped probabilistic graphical model (PGM) in full analogy

to the implicit shape model [33.44] (Sect. 33.2.3) or a special case of a tree-structured pictorial structures model [33.39] (Sect. 33.2.3). While this is less powerful than a full constellation model [33.41] in terms of representation, it has proven very effective in connection with a discriminative learning regime.

Second, the DPM is formulated as a mixture model with multiple components, where each component takes care of another aspect of the object class of interest, such as a particular viewpoint or a particular context in which the object occurs (e.g., a person sitting on a chair). Each component consists of a star-model with root and multiple part templates as described earlier, trained to respond the strongest to its dedicated aspect – this separation of concerns leads to a simpler learning problem than trying to handle all aspects with a single component.

For recognition, each of the DPM components produces a score as follows: first, all of the component's templates (root and parts) are convolved with the image in a sliding window fashion, as in the original HOG [33.47] work (Sect. 33.2.4), resulting in dense score maps over image locations and scales. Second, root and part scores are combined with spatial deformation terms that reflect the star-shaped spatial dependency structure, resulting in new score maps over locations and scales – note that this inference step can be carried out by max-product belief propagation, which is exact and efficient due to the star topology, and implemented as an instance of a generalized distance transform [33.59]. And third, for each image location and scale, the maximum response over all components is kept as the final detection score (which is then further processed by a standard, greedy nonmaxima suppression).

Learning

It has been argued that much of the DPM's excellent detection performance is due to the specific way in which parameters are learned from training data, in addition to the added expressive power in comparison to a single, rigid HOG template. This learning also constitutes the major difference between the DPM and earlier flexible part models such as the ISM [33.44] or the pictorial structures model [33.39] (Sect. 33.2.3) that are structurally equivalent, but less effective in terms of performance.

In contrast to these earlier models, where local part appearance and spatial layout models are trained separately, all DPM parameters are learned jointly, such that a single binary classification objective is optimized (distinguish between the object class of interest and background). Crucially, this includes the placement of parts relative to the root template as well as the selection of the mixture component that is deemed responsible; both are treated as latent variables at training (and test) time. As a consequence, all aspects of the DPM are optimized towards the single goal of object class detection, without being convoluted by potentially suboptimal choices of part placement based on appearance similarity (ISM) or semantic meaning (pictorial structures).

The joint optimization problem is phrased as a generalization of the linear SVM [33.21] formulation of a single HOG template, termed latent support vector machine (LSVM) [33.60]. It is based on the following intuition: assuming that the latent variables (part placements and mixture component selection) are observed, all model parameters (HOG template and part displacement weights) can be learned using standard linear SVM procedures (the resulting objective function is convex, assuming the use of a hinge loss). Conversely, assuming known model parameters, the optimal values of all hidden variables can be found exactly and efficiently (using the distance transform for part placement and exhaustive search over components). This intuition gives rise to an EM-like alternation of two steps, which can be shown to converge to a local optimum of the original, nonconvex objective function.

Naturally, a good initialization is needed in order to reach good local optima during learning. The DPM suggests a clustering of training examples based on aspect ratio, which has proven to work surprisingly well in practice (clustering based on appearance typically results in small improvements but is much more costly in terms of computation time). As for the HOG detector [33.47], bootstrapping (here called *hard negative mining*) allows to escape bad local optima.

While the concept of jointly optimizing part appearance and placement has already existed in the constellation model [33.41], the DPM achieves this goal by largely different technical means, resulting in better performance: (1) it uses sliding-window HOG templates with linear SVM classifiers instead of a generative model over local interest point descriptors to represent local appearance, and (2) it relaxes the fully connected probabilistic graphical model topology to a star, making exact inference tractable and efficient for learning. The DPM has achieved excellent performance on a wide variety of benchmark datasets, including PASCAL VOC (visual object class) [33.61] (Sect. 33.2.8).

Multiview and 3-D DPMs

While the original DPM [33.53] achieves remarkable performance on standard detection benchmarks, it is inherently limited to predicting 2-D bounding boxes. While this might be sufficient for some applications (such as counting the number of objects in an image), it

Avidbots Ex1016
Page 867 of 2154

is clearly not enough for others: suppose an autonomous driving scenario in which vehicles are to be tracked across multiple frames, and their spatial extent to be estimated in order to avoid collisions – here, a 3-D representation of each vehicle is required.

To that end, the DPM has been extended [33.56, 62, 63] to output richer object hypotheses than 2-D bounding boxes, in the form of additional viewpoint estimates (providing a better notion of 3-D object extent than a 2-D bounding box alone) and 3-D parts that are consistent across viewpoints (providing a basis for establishing correspondences across multiple frames in a sequence). Providing these richer outputs can be achieved by extending the original DPM in two different ways: first, the classification problem solved during learning can be reformulated as a structured output prediction problem [33.62]. In this way, the model can be optimized for the prediction of other quantities than a binary object-background label, such as 2-D bounding box overlap or angular viewpoint estimates, through the use of different loss functions. In its simplest variant, viewpoint estimation can be implemented by a binning of angles, and dedicating a single DPM mixture component to each angular bin. While this model is still inherently view-based, it has been shown to deliver excellent performance in both 2-D bounding box localization and viewpoint estimation (note that viewpoint labels have to be available during training in addition to bounding box annotations).

Second, instead of parameterizing the DPM parts in the image plane, an object-centric 3-D coordinate system can be used, such that parts from different mixture components refer to the same 3-D object portion (Fig. 33.6a). Part deformations are then governed by 3-D displacement distributions (Fig. 33.6b), and object as well as part appearance can be captured in a continuous appearance model (Fig. 33.7a). The result is a 3-D object class representation [33.63] that can synthesize previously unseen viewpoints at test time (by interpolating between a number of supporting viewpoints for which training data is available) and thus estimate viewpoints up to arbitrary granularity. The learning of this 3-D object class representation is facilitated by using 3-D computer-aided design (CAD) models of the object class of interest in addition to real-world images (Fig. 33.7b). During training, the CAD models serve both to establish a 3-D coordinate system and to generate artificial training data for different viewpoints through rendering. Figure 33.8 shows example detections obtained by the 3-D^{2}PM.

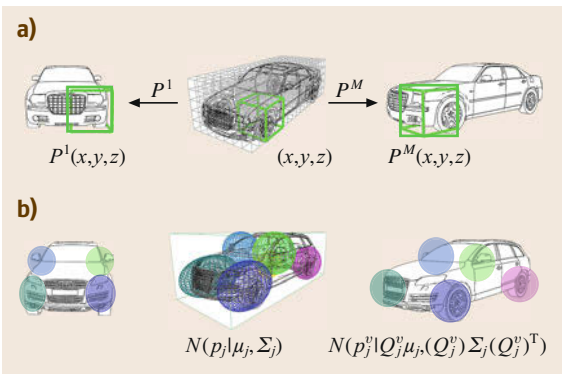


Fig.33.6a,b Part representation in the 3-D deformable part model (3-D DPM). **(a)** 3-D/2-D part coordinate systems; (after [33.62]) **(b)** 3-D/2-D part spatial distributions (after [33.63])

Computational Considerations

Interestingly, it has been realized that the major computational bottleneck of the original DPM [33.53] at recognition time is the convolution of HOG templates with image features (and not probabilistic graphical model inference as one might think). As a consequence, there have been attempts to minimize the number of HOG convolutions by various means, e.g., by part sharing parts across components and object classes [33.64], re-using parts of templates by decomposing them into an overcomplete set of dictionary elements [33.65] or cascaded detector evaluation [33.66]. Likewise, there have been attempts to reduce the training time for the DPM by exploiting HOG feature statistics to re-use computation across models [33.67, 68].

33.2.6 Convolutional Neural Networks

While artificial neural networks (ANNs) [33.69] had largely been neglected in favor of support vector machines, random forests, and other learners until around 2010, they have recently returned to the scene, with quite remarkable success. Among a number of artificial

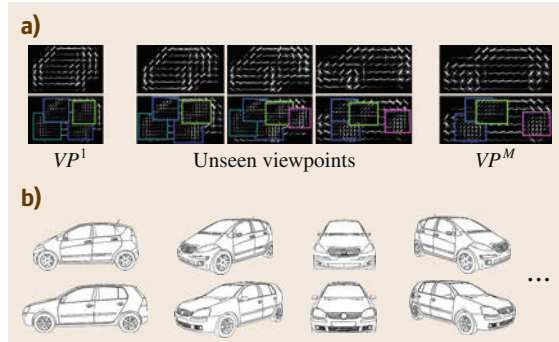


Fig.33.7a,b Continuous appearance model **(a)** in the 3-D^{2}PM, trained from rendered 3-D CAD models (after [33.63]) **(b)** and real images (not shown)

Avidbots Ex1016
Page 868 of 2154

neural network architectures that had been suggested already in the early days of computer vision, the convolutional neural network (CNN) [33.70] has emerged as a particularly promising candidate for becoming a standard tool for visual recognition tasks.

One of the key properties of the CNN is its specific structure (Fig. 33.9), which is both amenable to dealing with multidimensional input (such as a color image) and effective for preventing overfitting when learned from limited training data. As an ANN, a CNN defines an artificial neural network with multiple layers on an input image, such that image pixels provide the input to the neurons of the first layer, and the outputs of each layer of neurons provide the inputs to the next layer (the final layer can provide a task specific output, such as an encoding of different image classes). Now, the topology of the CNN is restricted in two specific ways: first, neurons in one layer can only be connected to a fixed spatial region of neurons in the preceding layer (the *receptive field*) – this dramatically reduces the total number of connections in the network and hence parameters of the model that have to be learned. And second, most layers of neurons (typically the lower layers) are formed by replicating local groups of neurons multiple times, including the weights associated with their connections to the preceding layer (*weight sharing*) – this further reduces the number of parameters of the model, and encourages the network to learn a representation for the input that is invariant to translation. As a result of their specific structure, CNNs are less prone to overfitting than fully connected ANNs, and can be successfully trained using gradient-based methods and back-propagation of errors.

CNNs for Image Classification

CNNs have recently achieved remarkable performance in large-scale image classification [33.71], fueled by the availability of large-scale datasets like ImageNet [33.33] that provide thousands of images per object class. The success of CNNs in image classification is mostly attributed to their ability to learn useful image representations end-to-end (*deep learning*) rather than relying on engineered features, such as SIFT [33.9] or HOG [33.47]. While the latter have been carefully designed and empirically tuned over many years, they are *general purpose* and not specifically adapted to the classification task at hand, and hence suboptimal. A recent study provides further insights into the operation of individual CNN components (hidden layers and classifiers) through visualization [33.74], resulting in a further boost in performance.

Region Proposals for Detection

Since the evaluation of a CNN is computationally expensive due to convolutions, a CNN is typically applied to preselected candidate regions rather than in a dense sliding window fashion for detection. Candidate regions can be generated either by a specialized detector trained for the object class of interest (such as HOG [33.47], Sect. 33.2.4) that is weaker but also much faster to evaluate than the CNN, or by a more general attention mechanism that generates bounding boxes that are likely to contain an object instance of *any* class. Among various methods that generate proposal regions with a high degree of *objectness* [33.75] ([33.76] for a recent discussion and analysis of region proposal methods), selective search [33.77] has been successfully used in



Fig. 33.8 3-D^{2}PM example detections (car, bicycle, human head, cell phone)

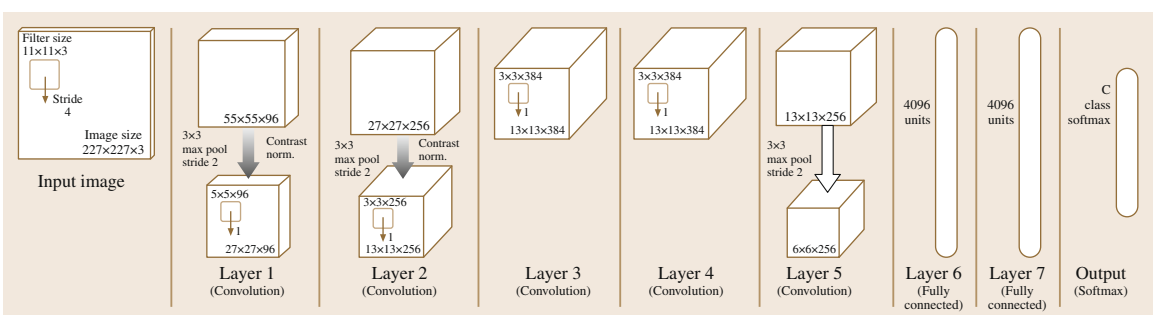


Fig. 33.9 The CNN model of (after [33.71]) in the implementation of Caffe (after [33.72, 73])

Avidbots Ex1016
Page 869 of 2154

connection with CNN-based, object class specific classifiers for detection [33.78]. Selective search is built upon a superpixel-based oversegmentation of an image, and suggests a greedy procedure to group adjacent superpixels into regions of high objectness. The resulting detector [33.79] has been shown to outperform previous results in generic object class detection by a large margin (notably outperforming the previous best results of the DPM [33.53], Sect. 33.2.5, on PASCAL VOC [33.61], Sect. 33.2.8). Curiously, it is not necessary to train this detector end-to-end for the task of object class detection for good performance – instead, the lower layers of the CNN can be pretrained for classification on ImageNet as in [33.71], and later fine-tuned for detection. Both the CNN pretrained on ImageNet and the resulting state-of-the-art object class detector are publicly available under the name *Caffe* [33.72] and likely to find their way into the toolboxes of many computer vision practitioners.

Structured Outputs

While artificial neural networks can represent any function in theory, effective learning of ANNs with more complex, structured outputs than single class labels or continuous values remains a challenge. Instead of learning a full network end-to-end, structured outputs are typically obtained by feeding the outputs of the last layer of neurons into an existing structured model, such as multiple body joint regressors for human body pose estimation [33.80].

33.2.7 Fine-Grained Categorization

In recent years, there has been a growing interest in distinguishing objects not only with respect to basic-level categories [33.5] (such as car, bicycle, and person), but also on a finer level of granularity (such as different car models or plant and animal species). This *fine-grained categorization* is deemed particularly challenging: the distinction between two categories is often determined by small visual differences (such as the shape of a car's headlight): these differences are easily outweighed by appearance differences due to change in lighting or viewpoint (a car looks largely different from the front and from the side), bearing the risk of overfitting to irrelevant details.

Feature Selection

A first strategy to identifying subtle discriminative features while at the same time preventing overfitting is to combine classical feature selection techniques with randomization, e.g., in the form of random forest classifiers [33.35, 81]. Here, randomization acts as a regularizer that prevents multiple trees to latch onto the same potentially irrelevant feature, hence reducing the risk of overfitting. Feature selection can also be performed with humans in the loop, using crowd-sourcing and gamification [33.82].

Pose Normalization

A second, orthogonal strategy in fin-grained categorization is to devise factored object class representations that explicitly capture different modes of variation in object appearance, thereby separating variations caused by category membership from variations caused by other factors, such as viewpoint changes. The main idea is to establish a common frame of reference for the extraction of visual features (e.g., a local coordinate system for the dense extraction of SIFT [33.9] features), such that the same object instance will always result in (roughly) the same visual features being extracted, irrespective of viewpoint, lighting, etc.

Specifically, such a frame of reference can be established by first detecting the object of interest in a test image and also localizing its semantic parts. Visual features are then extracted and described relative to the estimated object position and detected parts, ensuring that a feature on the beak of a bird will only ever be compared to a corresponding feature on the beak of another bird, and not on its tail (assuming perfect detection, viewpoint estimation, and part localization). This kind of part-based *pose normalization* has been demonstrated to be effective for bird species categorization [33.83, 84].

Instead of localizing object parts in 2-D, some works establish a common reference frame by estimating a rough 3-D geometry of the object of interest prior to feature extraction, in addition to estimating its viewpoint. This allows not only to describe feature locations relative to the object surface rather than the 2-D image plane (effectively performing the pooling in 3-D [33.36]), but also to rectify image patches with respect to the estimated surface normals, improving the robustness of the resulting feature descriptors with respect to small viewpoint changes. This strategy has proven effective in connection with a prebuilt bird model [33.35] as well as in connection with predicted 3-D CAD model matches for fine-grained car categorization [33.36].

33.2.8 Datasets

Since visual object class recognition approaches rely on machine learning algorithms at their core, they are very much dependent on the availability of sufficient training data – it has in fact been argued that major advances

in the field might have occurred in response of obtaining more data rather than by developing more powerful algorithms [33.85]. At the same time, datasets are of utmost importance in order to quantify and compare the performance of different methods under controlled conditions, and checking the sanity of newly developed algorithms. To that end, benchmark datasets for visual object class recognition typically consist of a collection of images split into training, validation, and test sets, plus a set of ground truth annotations, and often also evaluation scripts that measure the performance of a recognition algorithm relative to the provided annotations.

Since each dataset provides only a small sample of the visual world, there is an inherent danger of being biased [33.85] with respect to which images are selected to be part of a dataset (and which ones are not), and by what means images are captured. As a result, new recognition algorithms should always be tested and cross-trained on multiple datasets in order to prevent overfitting to the characteristics of a particular dataset rather than learning a concept of interest. The following paragraphs give a small, by no means complete, selection of some of today's influential datasets for visual object class recognition.

PASCAL VOC

The PASCAL visual object classes challenge [33.61] stands for a succession of datasets that have been released on a yearly basis from 2005 to 2012. Each dataset consists of multiple challenges (image classification, object class detection, segmentation, human body pose estimation) that are defined on a given set of images. Notably, the set of images is steadily increased, such that images of year n are re-used in year $n+1$. PASCAL VOC counteracts dataset overfitting by not releasing test set ground truth annotations to the public, but instead maintaining an evaluation server that runs a limited number of automated tests for a particular recognition algorithm. PASCAL VOC images have been equipped with additional annotations, e.g., in the form of aligned 3-D CAD models (3-D-PASCAL [33.86]).

SUN and COCO

The scene understanding (SUN) [33.87] and common objects in context (COCO) [33.88] datasets have been built in an effort to capture entire visual scenes rather than objects in isolation. They are meant as benchmarks for methods that reason jointly about multiple objects in a scene rather than performing independent classification or detection.

ImageNet

ImageNet [33.33] is one of the largest image datasets to date, consisting of over 14 million internet images that have been annotated with object labels from the WordNet [33.89] hierarchy. Like PASCAL VOC, it constitutes the basis for multiple challenges that are defined on the dataset, ranging from classification (1000 classes) over object class detection to fine-grained categorization. Due to its massive size, the collection of the ImageNet dataset has been performed mostly by means of crowd sourcing. ImageNet constitutes the basis for pretraining some of the most successful CNN-based image classifiers to date [33.71] (Sect. 33.2.6).

33.3 Discussion and Conclusions

In this chapter, we have discussed visual object class recognition as one of the most fundamental problems in computer vision, with a focus on presenting the most promising models available to practitioners today. The selection of methods that we have reviewed is by no means complete – in favor of going into some depth for the most common techniques, we have skipped over object representations based on manifold learning [33.90] as well as representations based on grammars [33.91, 92], to mention just a few.

As we have seen, practically all available models are inherently based on some notion of object parts that are governed by certain topological constraints. Naturally, the choice of which part-based model to pick is not an easy one, as it depends on many different factors that should be carefully considered.

33.3.1 Task

First of all, the appropriate choice of model is dependent on the application scenario and the resulting desired recognition output: different methods can generate outputs of different granularity, ranging from image-level class labels (basic bag-of-words, Sect. 33.2.1, spatial pyramid matching, Sect. 33.2.2) over 2-D bounding box locations (bag-of-words and spatial pyramid matching run in sliding window fashion, histogram of oriented gradient detectors, Sect. 33.2.4, flexible part models, Sect. 33.2.3, basic deformable part models, Sect. 33.2.5) to multiview detections and viewpoint estimates (multiview ISM, Sect. 33.2.3, 3-D DPM, Sect. 33.2.5).

Secondly, an application scenario typically comes with a certain set of object classes of interest, which

Avidbots Ex1016
Page 871 of 2154

are in turn characterized by rather different visual properties. Rigid objects (like cars or mugs) tend to be handled well by rigid templates (like histogram of oriented gradients, Sect. 33.2.4) or deformable part models (Sect. 33.2.5), while articulate objects (like cats and dogs) can be more reliably represented using looser spatial models like bag of words (Sect. 33.2.1), or by explicitly modeling their kinematic tree structure (pictorial structures, Sect. 33.2.3).

Finally, it should be noted that the distinction between object classes itself is quite arbitrary when drawing strict borders between any two classes. In reality, some classes are inherently more similar than others (e.g., dogs and horses are more similar than dogs and cars, since dogs and horses are both quadrupeds). Looking at multiple levels of object categorization rather than individual classes, it becomes a desired property that objects from the same super-ordinate category, such as quadrupeds, be classified as more similar than objects from different super-ordinate categories. In other words, an object categorization system should degrade gracefully: if the object itself is not correctly recognized, then we want it to be assigned at least to a *similar* category. We thus need a way of representing the relationship and grading similarities between object categories [33.93].

33.3.2 Data

A particular choice of model implies a certain amount of parameters that have to be learned and additional latent parameters that have to be inferred at test time. A model has to be chosen not only appropriate for the task (above) but also in accordance with the available data at hand in order to prevent over or under fitting.

33.3.3 Efficiency

While simple *feedforward* models like histograms of oriented gradients (Sect. 33.2.4) and bag-of-words (Sect. 33.2.1) are very efficient, modeling deformations and interdependence of parts (flexible part models, Sect. 33.2.3, deformable part models, Sect. 33.2.5) comes at the cost of inferring latent quantities in the model. Notably, this inference often has to be performed repeatedly during learning, underlining the importance of efficient inference procedures.

References

33.1 D. Marr: *Vision* (Freeman, San Francisco 1982)
33.2 D.H. Ballard, C.M. Brown: *Computer Vision* (Prentice Hall, Englewood Cliffs 1982)
33.3 R. Brown: How shall a thing be called?, Psychol. Rev. **65**, 14–21 (1958)
33.4 R. Brown: *Social Psychology* (Free, New York 1965)
33.5 E. Rosch, C. Mervis, W. Gray, D. Johnson, P. Boyes-Braem: Basic objects in natural categories, Cogn. Psychol. **8**, 382–439 (1976)
33.6 G. Lakoff: *Women, Fire, and Dangerous Things – What Categories Reveal About the Mind* (Univ. Chicago Press, Chicago 1987)
33.7 S. Dickinson, A. Leonardis, B. Schiele, M. Tarr: *Object Categorization: Computer and Human Vision Perspectives* (Cambridge Univ. Press, Cambridge 2009)
33.8 G. Csurka, C.R. Dance, L. Fan, J. Willarnowski, C. Bray: Visual categorization with bags of keypoints, Eur. Conf. Comput. Vis. (ECCV) (2004)
33.9 D. Lowe: Distinctive image features from scale-invariant keypoints, Int. J. Comput. Vis. **60**(2), 91–110 (2004)
33.10 K. Mikolajczyk, C. Schmid: A performance evaluation of local descriptors, IEEE Trans. Pattern Anal. Mach. Intell. **27**(10), 1615–1630 (2004)
33.11 J. Matas, O. Chum, M. Urban, T. Pajdla: Robust wide baseline stereo from maximally stable extremal regions, Image Vis. Comput. **22**(10), 761–767 (2004)
33.12 T. Tuytelaars: Dense interest points, IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR) (2010)
33.13 R.O. Duda, P.E. Hart, D.G. Stork: *Pattern Classification* (Wiley, New York 2000)
33.14 F. Jurie, B. Triggs: Creating efficient codebooks for visual recognition, 10th IEEE Int. Conf. Comput. Vis. (ICCV) (2005)
33.15 B. Schiele, J.L. Crowley: Recognition without correspondence using multidimensional receptive field histograms, Int. J. Comput. Vis. **36**(1), 31–52 (2000)
33.16 T. Hofmann: Probabilistic latent semantic indexing, Proc. 22nd Annu. Int. ACM SIGIR Conf. Res. Dev. Inf. Retr. (1999)
33.17 D.M. Blei, A.Y. Ng, M.I. Jordan: Latent dirichlet allocation, J. Mach. Learn. Res. **3**, 983–1022 (2003)
33.18 A.P. Dempster, N.M. Laird, D.B. Rubin: Maximum likelihood from incomplete data via the EM algorithm, J. R. Statist. Soc. B **39**, 1–38 (1977)
33.19 Z. Shi, Y. Yang, T.M. Hospedales, T. Xiang: Weakly supervised learning of objects, attributes and their associations, Eur. Conf. Comput. Vis. (ECCV) (2014) pp. 472–487
33.20 S. Lazebnik, C. Schmid, J. Ponce: Spatial pyramid matching. In: *Object Categorization*, ed. by S. Dickinson, A. Leonardis, B. Schiele, M. Tarr (Cambridge Univ. Press, Cambridge 2009) pp. 401–415
33.21 C. Cortes, V. Vapnik: Support-vector networks, Mach. Learn. **20**(3), 273–297 (1995)
33.22 S. Maji, A.C. Berg, J. Malik: Classification using intersection kernel support vector machines is efficient, IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR) (2008)
33.23 Y. Freund, R.E. Schapire: A decision-theoretic generalization of on-line learning and an application

Avidbots Ex1016
Page 872 of 2154

to boosting, J. Comput. Syst. Sci. **55**(1), 119–139 (1997)
33.24 L. Breiman: Random forests, Mach. Learn. **45**(1), 5–32 (2001)
33.25 T. Lindeberg: Feature detection with automatic scale selection, Int. J. Comput. Vis. **30**(2), 79–116 (1998)
33.26 C.H. Lampert, M.B. Blaschko, T. Hofmann: Efficient subwindow search: A branch and bound framework for object localization, IEEE Trans. Pattern Anal. Mach. Intell. **31**(12), 2129–2142 (2009)
33.27 S. Lazebnik, C. Schmid, J. Ponce: Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories, IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR) (2006)
33.28 V. Ferrari, L. Fevrier, F. Jurie, C. Schmid: Groups of adjacent contour segments for object detection, IEEE Trans. Pattern Anal. Mach. Intell. **30**(1), 36–51 (2008)
33.29 E. Shechtman, M. Irani: Matching local self-similarities across images and videos, IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR) (2007)
33.30 F. Perronnin, C. Dance: Fisher kernels on visual vocabularies for image categorization, IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR) (2007)
33.31 J. Yang, K. Yu, Y. Gong, T. Huang: Linear spatial pyramid matching using sparse coding for image classification, IEEE Conf. Comput. Vis. Pattern Recogn. (CVPR) (2009)
33.32 J. Wang, J. Yang, K. Yu, F. Lv, T. Huang, Y. Gong: Locality-constrained linear coding for image classification, IEEE Conf. Comput. Vis. Pattern Recogn. (CVPR) (2010)
33.33 J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, L. Fei-Fei: ImageNet: A large-scale hierarchical image database, IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR) (2009)
33.34 N. Kruger, P. Janssen, S. Kalkan, M. Lappe, A. Leonardis, J. Piater, A.J. Rodriguez-Sanchez, L. Wiskott: Deep hierarchies in the primate visual cortex: What can we learn for computer vision?, IEEE Trans. Pattern Anal. Mach. Intell. **35**(8), 1847–1871 (2013)
33.35 R. Farrell, O. Oza, N. Zhang, V.I. Morariu, T. Darrell, L.S. Davis: Birdlets: Subordinate categorization using volumetric primitives and pose-normalized appearance, IEEE Int. Conf. Comput. Vis. (ICCV) (2011)
33.36 J. Krause, M. Stark, J. Deng, L. Fei-Fei: 3d object representations for fine-grained categorization, 4th Int. IEEE Workshop 3D Represent. Recognit. (3dRR-13), Sydney (2013)
33.37 M. Varma, D. Ray: Learning the discriminative power-invariance trade-off, IEEE Conf. Comput. Vis. Pattern Recogn. (CVPR) (2007)
33.38 A. Vedaldi, V. Gulshan, M. Varma, A. Zisserman: Multiple kernels for object detection, IEEE Int. Conf. Comput. Vis. (ICCV) (2009)
33.39 M.A. Fischler, R.A. Elschlager: The representation and matching of pictorial structures, IEEE Trans. Comput. **22**(1), 67–92 (1973)
33.40 M. Weber, M. Welling, P. Perona: Unsupervised learning of models for recognition, Eur. Conf. Comput. Vis. (ECCV) (2000)
33.41 R. Fergus, P. Perona, A. Zisserman: Object class recognition by unsupervised scale-invariant learning, IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR) (2003)
33.42 M. Stark, M. Goesele, B. Schiele: A shape-based object class model for knowledge transfer, IEEE Int. Conf. Comput. Vis. (ICCV) (2009)
33.43 L. Fei-Fei, R. Fergus, P. Perona: Learning generative visual models from few training examples: An incremental Bayesian approach tested on 101 object categories, IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR) (2004) pp. 178–186
33.44 B. Leibe, A. Leonardis, B. Schiele: Robust object detection by interleaving categorization and segmentation, Int. J. Comput. Vis. **77**(1–3), 259–289 (2008)
33.45 S. Maji, J. Malik: Object detection using a max-margin Hough transform, IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR) (2009)
33.46 L. Bourdev, J. Malik: Poselets: Body part detectors trained using 3D human pose annotations, IEEE Int. Conf. Comput. Vis. (ICCV) (2009)
33.47 N. Dalal, B. Triggs: Histograms of oriented gradients for human detection, IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR) (2005) pp. 886–893
33.48 B. Leibe, E. Seemann, B. Schiele: Pedestrian detection in crowded scenes, IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Washington (2005) pp. 878–885
33.49 A. Thomas, V. Ferrari, B. Leibe, T. Tuytelaars, B. Schiele, L. Van Gool: Towards multi-view object class detection, IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR) (2006)
33.50 M. Arie-Nachimson, R. Basri: Constructing implicit 3D shape models for pose estimation, IEEE Int. Conf. Comput. Vis. (ICCV) (2009)
33.51 P.F. Felzenszwalb, D.P. Huttenlocher: Efficient matching of pictorial structures, IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR) (2000)
33.52 M. Andriluka, S. Roth, B. Schiele: Pictorial structures revisited: People detection and articulated pose estimation, IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR) (2009)
33.53 P.F. Felzenszwalb, R. Girshick, D. McAllester, D. Ramanan: Object detection with discriminatively trained part based models, IEEE Trans. Pattern Anal. Machin. Intell. **32**(9), 1627–1645 (2010)
33.54 F. Wang, Y. Li: Beyond physical connections: Tree models in human pose estimation, IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR) (2013) pp. 596–603
33.55 M. Sun, M. Telaprolu, H. Lee, S. Savarese: An efficient branch-and-bound algorithm for optimal human pose estimation, IEEE Conf. Comput. Vis. Pattern Recogn. (CVPR) (2012)
33.56 P. Bojan, M. Stark: Multi-view and 3D deformable part models, IEEE Trans. Pattern Anal. Mach. Intell. (TPAMI) **37**(11), 2232–2245 (2015)
33.57 N. Dalal, B. Triggs, C. Schmid: Human detection using oriented histograms of flow and appearance, Eur. Conf. Comput. Vis. (ECCV) (2006)
33.58 G. Bradski: Opencv. http://opencv.org/ (July 09, 2015)

Avidbots Ex1016
Page 873 of 2154

33.59 P.F. Felzenszwalb, D.P. Huttenlocher: *Distance transforms of sampled functions, Technical Report 1963* (Cornell Univ., Ithaca 2004)

33.60 C.-N.J. Yu, T. Joachims: Learning structural SVMs with latent variables, ACM Proc. 26th Annu. Int. Conf. Mach. Learn., New York (2009) pp. 1169–1176

33.61 M. Everingham, L. Gool, C.K. Williams, J. Winn, A. Zisserman: The PASCAL visual object classes (VOC) challenge, Int. J. Comput. Vis. **88**(2), 303–338 (2010)

33.62 B. Pepik, M. Stark, P. Gehler, B. Schiele: Teaching 3d geometry to deformable part models, IEEE Conf. Comput. Vis. Pattern Recogn. (CVPR) (2012)

33.63 B. Pepik, P. Gehler, M. Stark, B. Schiele: $3D^2PM$–3D deformable part models, Eur. Conf. Comput. Vis. (ECCV) (2012)

33.64 P. Ott, M. Everingham: Shared parts for deformable part-based models, IEEE Comput. Vis. Pattern Recognit. (CVPR) (2011)

33.65 H.O. Song, S. Zickler, T. Althoff, R. Girshick, M. Fritz, C. Geyer, P. Felzenszwalb, T. Darrell: Sparselet models for efficient multiclass object detection, Eur. Conf. Comput. Vis. (ECCV) (2012)

33.66 P. Felzenszwalb, R. Girshick, D. McAllester: Cascade object detection with deformable part models, IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR) (2010)

33.67 T. Gao, M. Stark, D. Koller: What makes a good detector? – structured priors for learning from few examples, Eur. Conf. Comput. Vis. (ECCV) (2012)

33.68 B. Hariharan, J. Malik, D. Ramanan: Discriminative decorrelation for clustering and classification, Eur. Conf. Comput. Vis. (ECCV) (2012)

33.69 S. Haykin: *Neural Networks: A Comprehensive Foundation*, 2nd edn. (Prentice Hall, Upper Saddle River 1998)

33.70 Y. LeCun, B. Boser, J.S. Denker, D. Henderson, R.E. Howard, W. Hubbard, L.D. Jackel: Backpropagation applied to handwritten zip code recognition, Neural Comput. **1**(4), 541–551 (1989)

33.71 A. Krizhevsky, I. Sutskever, G.E. Hinton: Imagenet classification with deep convolutional neural networks, Adv. Neural Inform. Process. Syst. **25**, 1097–1105 (2012)

33.72 Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, T. Darrell: Caffe: Convolutional architecture for fast feature embedding, http://caffe.berkeleyvision.org/ (arXiv preprint arXiv:1408.5093) (2014)

33.73 J. Hosang, M. Omran, R. Benenson, B. Schiele: Taking a deeper look at pedestrians, IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR) (2015)

33.74 M.D. Zeiler, R. Fergus: Visualizing and understanding convolutional networks, Eur. Conf. Comput Vis. (ECCV) (2014)

33.75 B. Alexe, T. Deselares, V. Ferrari: Measuring the objectness of image windows, IEEE Trans. Pattern Anal. Mach. Intell. **34**(11), 2189–2202 (2012)

33.76 J. Hosang, R. Benenson, B. Schiele: How good are detection proposals, really?, 25th Br. Mach. Vis. Conf. (BMVC) (2014)

33.77 K.E.A. van de Sande, J.R.R. Uijlings, T. Gevers, A.W.M. Smeulders: Segmentation as selective search for object recognition, IEEE Int. Conf. Comput. Vis. (ICCV) (2013)

33.78 B. Hariharan, P. Arbeláez, R. Girshick, J. Malik: Simultaneous detection and segmentation, Eur. Conf. Comput. Vis. (ECCV) (2014)

33.79 R. Girshick, J. Donahue, T. Darrell, J. Malik: Rich feature hierarchies for accurate object detection and semantic segmentation, IEEE Conf. Comput. Vis. Pattern Recogn. (CVPR) (2014)

33.80 A. Toshev, C. Szegedy: Deeppose: Human pose estimation via deep neural networks, IEEE Conf. Comput. Vis. Pattern Recogn. (CVPR) (2014)

33.81 J. Deng, J. Krause, M. Stark, L. Fei-Fei: Leveraging the wisdom of the crowd for fine-grained recognition, IEEE Trans. Pattern Anal. Mach. Intel. (2015)

33.82 J. Deng, J. Krause, L. Fei-Fei: Fine-grained crowdsourcing for fine-grained recognition, IEEE Conf. Comput. Vis. Pattern Recogn. (CVPR) (2013)

33.83 N. Zhang, R. Farrell, T. Darrell: Pose pooling kernels for sub-category recognition, IEEE Conf. Comput. Vis. Pattern Recogn. (CVPR) (2012)

33.84 T. Berg, P.N. Belhumeur: Poof: Part-based one-vs-one features for fine-grained categorization, face verification, and attribute estimation, IEEE Conf. Comput. Vis. Pattern Recogn. (CVPR) (2013)

33.85 A. Torralba, A.A. Efros: Unbiased look at dataset bias, IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR) (2011)

33.86 Y. Xiang, R. Mottaghi, S. Savarese: Beyond pascal: A benchmark for 3d object detection in the wild, IEEE Winter Conf. Appl. Comput. Vis. (WACV) (2014)

33.87 J. Xiao, J. Hays, K. Ehinger, A. Oliva, A. Torralba: Sun database: Large-scale scene recognition from abbey to zoo, IEEE Conf. Comput. Vis. Pattern Recogn. (CVPR) (2010)

33.88 T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, C.L. Zitnick: Microsoft coco: Common objects in context, Lect. Notes Comput. Sci. **8693**, 740–755 (2014)

33.89 G.A. Miller: Wordnet: A lexical database for english, ACM Communication **38**(11), 39–41 (1995)

33.90 M.A. Turk, A.P. Pentland: Face recognition using eigenfaces, IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR) (1991) pp. 586–591

33.91 Z.W. Tu, X.R. Chen, A.L. Yuille, S.C. Zhu: Image parsing: Unifying segmentation, detection and recognition, Int. J. Comput. Vis. **63**(2), 113–140 (2005)

33.92 L. Zhu, Y. Chen, A.L. Yuille: Unsupervised learning of probabilistic Grammar–Markov models for object categories, IEEE Trans. Pattern Anal. Machin. Intell. **31**(1), 114–128 (2009)

33.93 J. Deng, J. Krause, A. Berg, L. Fei-Fei: Hedging your bets: Optimizing accuracy-specificity trade-offs in large scale visual recognition, IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Providence (2012)

Avidbots Ex1016
Page 874 of 2154

Multimedia Contents

34. Visual Servoing

François Chaumette, Seth Hutchinson, Peter Corke

This chapter introduces visual servo control, using computer vision data in the servo loop to control the motion of a robot. We first describe the basic techniques that are by now well established in the field. We give a general overview of the formulation of the visual servo control problem, and describe the two archetypal visual servo control schemes: image-based and pose-based visual servo control. We then discuss performance and stability issues that pertain to these two schemes, motivating advanced techniques. Of the many advanced techniques that have been developed, we discuss two-and-a-half-dimensional (2.5-D), hybrid, partitioned, and switched approaches. Having covered a variety of control schemes, we deal with target tracking and controlling motion directly in the joint space and extensions to underactuated ground and aerial robots. We conclude by describing applications of visual servoing in robotics.

34.1 **The Basic Components of Visual Servoing** 842
34.2 **Image-Based Visual Servo** 843
34.2.1 The Interaction Matrix 843
34.2.2 Approximating the Interaction Matrix 844
34.2.3 A Geometrical Interpretation of IBVS 846
34.2.4 Stability Analysis 847
34.2.5 IBVS with a Stereo Vision System 848
34.2.6 IBVS with Cylindrical Coordinates of Image Points 849
34.2.7 IBVS with Other Geometrical Features 849
34.2.8 Non-Perspective Cameras 850
34.2.9 Direct Estimation 850
34.3 **Pose-Based Visual Servo** 851
34.4 **Advanced Approaches** 854
34.4.1 Hybrid VS 854
34.4.2 Partitioned VS 855
34.5 **Performance Optimization and Planning** 856
34.5.1 Optimal Control and Redundancy Framework 856
34.5.2 Switching Schemes 856
34.5.3 Feature Trajectory Planning 857
34.6 **Estimation of 3-D Parameters** 858
34.7 **Determining s^* and Matching Issues** 859
34.8 **Target Tracking** 859
34.9 **Eye-in-Hand and Eye-to-Hand Systems Controlled in the Joint Space** 860
34.10 **Under Actuated Robots** 861
34.11 **Applications** 863
34.12 **Conclusions** 863
Video-References 863
References 863

Visual servo (VS) *control* refers to the use of computer vision data to control the motion of a robot. The vision data may be acquired from a camera that is mounted directly on a robot manipulator or on a mobile robot, in which case motion of the robot induces camera motion, or the camera can be fixed in the workspace so that it can observe the robot motion from a stationary configuration. Other configurations can be considered, such as

Avidbots Ex1016
Page 875 of 2154

for instance several cameras mounted on pan–tilt heads observing the robot motion. The mathematical development of all these cases is similar, and in this chapter we will focus primarily on the former, so-called *eye-in-hand*, case.

Visual servo control relies on techniques from image processing, computer vision, and control theory. In the present chapter, we will deal primarily with the issues from control theory, making connections to previous chapters when appropriate.

34.1 The Basic Components of Visual Servoing

The aim of all vision-based control schemes is to minimize an error $\boldsymbol{e}(t)$, which is typically defined by

$$\boldsymbol{e}(t) = \boldsymbol{s}(\boldsymbol{m}(t), \boldsymbol{a}) - \boldsymbol{s}^* \,. \tag{34.1}$$

This formulation is quite general, and it encompasses a wide variety approaches, as we will see below. The parameters in (34.1) are defined as follows. The vector $\boldsymbol{m}(t)$ is a set of image measurements (e.g., the image coordinates of interest points, or the parameters of a set of image lines or segments). These image measurements are used to compute a vector of k visual features, $\boldsymbol{s}(\boldsymbol{m}(t), \boldsymbol{a})$, in which $\boldsymbol{a}$ is a set of parameters that represent potential additional knowledge about the system (e.g., true or approximate camera intrinsic parameters or a model of the object to be tracked). The vector $\boldsymbol{s}^*$ contains the desired values of the features. Note that the order of the desired and actual values in (34.1) is reversed with respect to the common convention for feedback control systems.

For now, we consider the case of a fixed goal pose and a motionless target, i. e., $\boldsymbol{s}^*$ is constant, and changes in $\boldsymbol{s}$ depend only on camera motion. Further, we consider here the case of controlling the motion of a camera with six degrees of freedom (e.g., a camera attached to the end effector of a six-degree-of-freedom arm). We will treat more general cases in later sections.

Visual servoing schemes mainly differ in the way that $\boldsymbol{s}$ is designed. In Sects. 34.2 and 34.3, we describe classical approaches, including image-based visual servo control (IBVS), in which $\boldsymbol{s}$ consists of a set of features that are immediately available in the image, and pose-based visual servo control (PBVS), in which $\boldsymbol{s}$ consists of a pose, which must be estimated from image measurements. Note that in the older visual servoing literature PBVS is named position-based, rather than pose-based, visual servoing [34.1, 2]. We also present in Sect. 34.4 several more-advanced methods.

Once $\boldsymbol{s}$ is selected, the design of the control scheme can be quite simple. Perhaps the most straightforward approach is to design a velocity controller. To do this, we require the relationship between the time variation of $\boldsymbol{s}$ and the camera velocity. Let the spatial velocity of the camera be denoted by $\boldsymbol{v}_c = (\boldsymbol{v}_c, \boldsymbol{\omega}_c)$ where $\boldsymbol{v}_c$ is the instantaneous linear velocity of the origin of the camera frame and $\boldsymbol{\omega}_c$ is the instantaneous angular velocity of the camera frame. The relationship between $\dot{\boldsymbol{s}}$ and $\boldsymbol{v}_c$ is given by

$$\dot{\boldsymbol{s}} = \mathbf{L}_{\boldsymbol{s}} \boldsymbol{v}_c \,, \tag{34.2}$$

in which $\mathbf{L}_{\boldsymbol{s}} \in \mathbb{R}^{k\times 6}$ is called the *interaction matrix* related to $\boldsymbol{s}$ [34.3]. The term *feature Jacobian* is also used somewhat interchangeably in the visual servo literature [34.2], but in the present chapter we will use this latter term to relate the time variation of the features to the robot's joint velocity (Sect. 34.8).

Using (34.1) and (34.2) we immediately obtain the relationship between camera velocity and the time variation of the error

$$\dot{\boldsymbol{e}} = \mathbf{L}_{\boldsymbol{e}} \boldsymbol{v}_c \,, \tag{34.3}$$

where $\mathbf{L}_{\boldsymbol{e}} = \mathbf{L}_{\boldsymbol{s}}$. Considering $\boldsymbol{v}_c$ as the input to the robot controller, and if we would like, for instance, to design for an exponential and decoupled decrease of the error (i. e., $\dot{\boldsymbol{e}} = -\lambda \boldsymbol{e}$) then using (34.3) we obtain as controller

$$\boldsymbol{v}_c = -\lambda \mathbf{L}_{\boldsymbol{e}}^{+} \boldsymbol{e} \,, \tag{34.4}$$

where $\mathbf{L}_{\boldsymbol{e}}^{+} \in \mathbb{R}^{6\times k}$ is the Moore–Penrose pseudo-inverse of $\mathbf{L}_{\boldsymbol{e}}$, that is, $\mathbf{L}_{\boldsymbol{e}}^{+} = (\mathbf{L}_{\boldsymbol{e}}^{\mathrm{T}} \mathbf{L}_{\boldsymbol{e}})^{-1} \mathbf{L}_{\boldsymbol{e}}^{\mathrm{T}}$ when $k \geq 6$ and $\mathbf{L}_{\boldsymbol{e}}$ is of full rank 6. When $k = 6$, if $\det \mathbf{L}_{\boldsymbol{e}} \neq 0$ it is possible to invert $\mathbf{L}_{\boldsymbol{e}}$, giving the control $\boldsymbol{v}_c = -\lambda \mathbf{L}_{\boldsymbol{e}}^{-1} \boldsymbol{e}$. When $k \leq 6$ and $\mathbf{L}_{\boldsymbol{e}}$ is of full rank k, $\mathbf{L}_{\boldsymbol{e}}^{+}$ is given by

$$\mathbf{L}_{\boldsymbol{e}}^{+} = \mathbf{L}_{\boldsymbol{e}}^{\mathrm{T}} \left(\mathbf{L}_{\boldsymbol{e}} \mathbf{L}_{\boldsymbol{e}}^{\mathrm{T}} \right)^{-1} \,.$$

When $\mathbf{L}_{\boldsymbol{e}}$ is not full rank, the numerical value of $\mathbf{L}_{\boldsymbol{e}}^{+}$ can be obtained from the singular value decomposition of $\mathbf{L}_{\boldsymbol{e}}$. In all cases, control scheme (34.4) allows $\|\dot{\boldsymbol{e}} - \lambda \mathbf{L}_{\boldsymbol{e}} \mathbf{L}_{\boldsymbol{e}}^{+} \boldsymbol{e}\|$ and $\|\boldsymbol{v}_c\|$ to be minimal. Note that the desired behavior $\dot{\boldsymbol{e}} = -\lambda \boldsymbol{e}$ is obtained only when $\mathbf{L}_{\boldsymbol{e}} \mathbf{L}_{\boldsymbol{e}}^{+} = \mathbf{I}_k$, where $\mathbf{I}_k$ is the $k \times k$ identity matrix, that is, only when $\mathbf{L}_{\boldsymbol{e}}$ if of full rank k, $k \leq 6$.

In real visual servo systems, it is impossible to know perfectly in practice either $\mathbf{L}_{\boldsymbol{e}}$ or $\mathbf{L}_{\boldsymbol{e}}^{+}$. So an approximation or an estimation of one of these two matrices must

Avidbots Ex1016
Page 876 of 2154

be realized. In the sequel, we denote both the pseudo-inverse of the approximation of the interaction matrix and the approximation of the pseudo-inverse of the interaction matrix by the symbol $\widehat{\mathbf{L}_e^+}$. Using this notation, the control law is in fact

$$\boldsymbol{v}_c = -\lambda \widehat{\mathbf{L}_e^+} \boldsymbol{e} = -\lambda \widehat{\mathbf{L}_s^+} (\boldsymbol{s} - \boldsymbol{s}^*) \,. \tag{34.5}$$

Closing the loop and assuming that the robot controller is able to realize perfectly $\boldsymbol{v}_c$, that is inserting (34.5) into (34.3), we obtain

$$\dot{\boldsymbol{e}} = -\lambda \mathbf{L}_e \widehat{\mathbf{L}_e^+} \boldsymbol{e} \,. \tag{34.6}$$

This equation characterizes the actual behavior of the closed-loop system, which is different from the desired one ($\dot{\boldsymbol{e}} = -\lambda \boldsymbol{e}$) when ever

$$\mathbf{L}_e \widehat{\mathbf{L}_e^+} \neq \mathbf{I}_k \,.$$

It is also the basis of the stability analysis of the system using Lyapunov theory.

What we have presented above is the basic design implemented by most visual servo controllers. All that remains is to fill in the details. How should $\boldsymbol{s}$ be chosen? What then is the form of $\mathbf{L}_s$? How should we estimate $\widehat{\mathbf{L}_e^+}$? What are the performance characteristics of the resulting closed-loop system? These questions are addressed in the remainder of the chapter. We first describe the two basic approaches, IBVS and PBVS, whose principles were proposed more than 20 years ago [34.1]. We then present more-recent approaches that have improved their performance.

34.2 Image-Based Visual Servo

Traditional image-based control schemes [34.1, 4] use the image-plane normalised coordinates of a set of points to define the vector $\boldsymbol{s}$. The image measurements $\boldsymbol{m}$ are usually the pixel coordinates of the set of image points (although this is not the only possible choice), and the parameters $\boldsymbol{a}$ in the definition of $\boldsymbol{s} = \boldsymbol{s}(\boldsymbol{m}, \boldsymbol{a})$ in (34.1) are nothing but the camera intrinsic parameters to go from image measurements expressed in pixels to the features.

34.2.1 The Interaction Matrix

A three-dimensional world point with coordinates $\boldsymbol{X} = (X, Y, Z)$ in the camera frame projects into the image plane of a conventional perspective camera as a two-dimensional point with normalised coordinates $\boldsymbol{x} = (x, y)$. More precisely we have

$$\begin{cases} x &= \dfrac{X}{Z} = \dfrac{u - c_u}{f\alpha} \\ y &= \dfrac{Y}{Z} = \dfrac{v - c_v}{f} \end{cases} , \tag{34.7}$$

where $\boldsymbol{m} = (u, v)$ gives the coordinates of the image point expressed in pixel units, and $\boldsymbol{a} = (c_u, c_v, f, \alpha)$ is the set of camera intrinsic parameters as defined in Chap. 32: c_u and c_v are the coordinates of the principal point, f is the focal length, and α is the ratio of the pixel dimensions. The intrinsic parameter β defined in Chap. 32 has been assumed to be 0 here. In this case, we take $\boldsymbol{s} = \boldsymbol{x} = (x, y)$, the image plane coordinates of the point. The details of the imaging geometry and perspective projection can be found in many computer vision texts, including [34.5–7].

Taking the time derivative of the projection equations (34.7), we obtain

$$\begin{cases} \dot{x} &= \dfrac{\dot{X}}{Z} - \dfrac{X\dot{Z}}{Z^2} = \dfrac{\dot{X} - x\dot{Z}}{Z} \\ \dot{y} &= \dfrac{\dot{Y}}{Z} - \dfrac{Y\dot{Z}}{Z^2} = \dfrac{\dot{Y} - y\dot{Z}}{Z} \end{cases} . \tag{34.8}$$

We can relate the velocity of the 3-D point to the camera spatial velocity using the well-known equation

$$\dot{\boldsymbol{X}} = -\boldsymbol{v}_c - \boldsymbol{\omega}_c \times \boldsymbol{X} \Leftrightarrow \begin{cases} \dot{X} = -v_x - \omega_y Z + \omega_z Y \\ \dot{Y} = -v_y - \omega_z X + \omega_x Z \\ \dot{Z} = -v_z - \omega_x Y + \omega_y X \,, \end{cases} \tag{34.9}$$

where $\boldsymbol{v}_c = (v_x, v_y, v_z)$ and $\boldsymbol{\omega}_c = (\omega_x, \omega_y, \omega_z)$. Inserting (34.9) into (34.8), grouping terms, and using (34.7) we obtain

$$\begin{cases} \dot{x} = \dfrac{-v_x}{Z} + \dfrac{x v_z}{Z} + xy\omega_x - (1 + x^2)\omega_y + y\omega_z \\ \dot{y} = \dfrac{-v_y}{Z} + \dfrac{y v_z}{Z} + (1 + y^2)\omega_x - xy\omega_y - x\omega_z \,, \end{cases} \tag{34.10}$$

which can be written

$$\dot{\boldsymbol{x}} = \mathbf{L}_{\boldsymbol{x}} \boldsymbol{v}_c \,, \tag{34.11}$$

where the interaction matrix $\mathbf{L}_x$ is given by

$$\mathbf{L}_x = \begin{pmatrix} \frac{-1}{Z} & 0 & \frac{x}{Z} & xy & -(1+x^2) & y \\ 0 & \frac{-1}{Z} & \frac{y}{Z} & 1+y^2 & -xy & -x \end{pmatrix} . \tag{34.12}$$

In the matrix $\mathbf{L}_x$, the value Z is the depth of the point relative to the camera frame. Therefore, any control scheme that uses this form of the interaction matrix must estimate or approximate the value of Z. Similarly, the camera intrinsic parameters are involved in the computation of x and y. Thus $\mathbf{L}_x^+$ cannot be *directly* used in (34.4), and an estimation or an approximation $\widehat{\mathbf{L}_x^+}$ must be used, as in (34.5). We discuss this in more detail below.

To control the six degrees of freedom, at least three points are necessary (i. e., we require $k \geq 6$). If we use the feature vector $\boldsymbol{x} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3)$, by merely stacking interaction matrices for three points we obtain

$$\mathbf{L}_x = \begin{pmatrix} \mathbf{L}_{\boldsymbol{x}_1} \\ \mathbf{L}_{\boldsymbol{x}_2} \\ \mathbf{L}_{\boldsymbol{x}_3} \end{pmatrix} .$$

In this case, there will exist some configurations for which $\mathbf{L}_x$ is singular [34.8]. Furthermore, there exist four distinct camera poses for which $\boldsymbol{e} = \mathbf{0}$, i. e., four global minima exist for the error function $\|\boldsymbol{e}\|$, and it is impossible to differentiate them [34.9]. For these reasons, more than three points are usually considered.

34.2.2 Approximating the Interaction Matrix

There are several choices available for constructing the estimate $\widehat{\mathbf{L}_e^+}$ to be used in the control law. One popular scheme is of course to choose

$$\widehat{\mathbf{L}_e^+} = \mathbf{L}_e^+$$

if $\mathbf{L}_e = \mathbf{L}_x$ is known, that is if the current depth Z of each point is available [34.2]. In practice, these parameters must be estimated at each iteration of the control scheme. The basic IBVS methods use classical pose-estimation methods (Chap. 32 and the beginning of Sect. 34.3). Another popular approach is to choose

$$\widehat{\mathbf{L}_e^+} = \mathbf{L}_{e^*}^+$$

where $\mathbf{L}_{e^*}$ is the value of $\mathbf{L}_e$ for the desired position $\boldsymbol{e} = \boldsymbol{e}^* = 0$ [34.3]. In this case, $\widehat{\mathbf{L}_e^+}$ is constant, and only the desired depth of each point has to be set, which means no varying 3-D parameters have to be estimated during the visual servo. Finally, the choice

$$\widehat{\mathbf{L}_e^+} = \left(\frac{\mathbf{L}_e}{2} + \frac{\mathbf{L}_{e^*}}{2} \right)^+$$

has been proposed [34.10]. Since $\mathbf{L}_e$ is involved in this method, the current depth of each point also has to be available.

We illustrate the behavior of these control schemes with an example. The goal is to position the camera so that it observes a square centered in the image (Fig. 34.1). We define $\boldsymbol{s}$ to include the x and y coordinates of the four points forming the square. Note

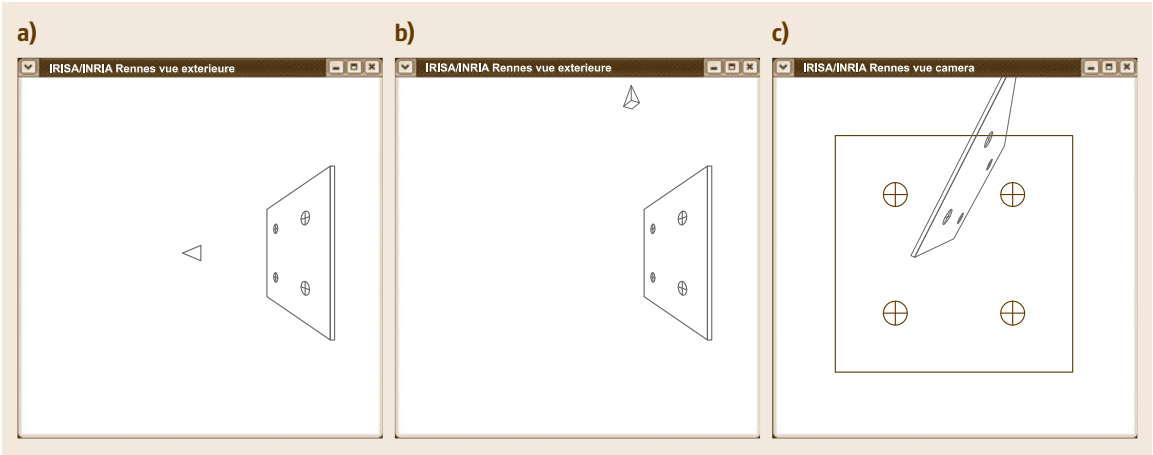


Fig.34.1a–c Example of positioning task: **(a)** the desired camera pose with respect to a simple target, **(b)** the initial camera pose, and **(c)** the corresponding initial and desired image of the target

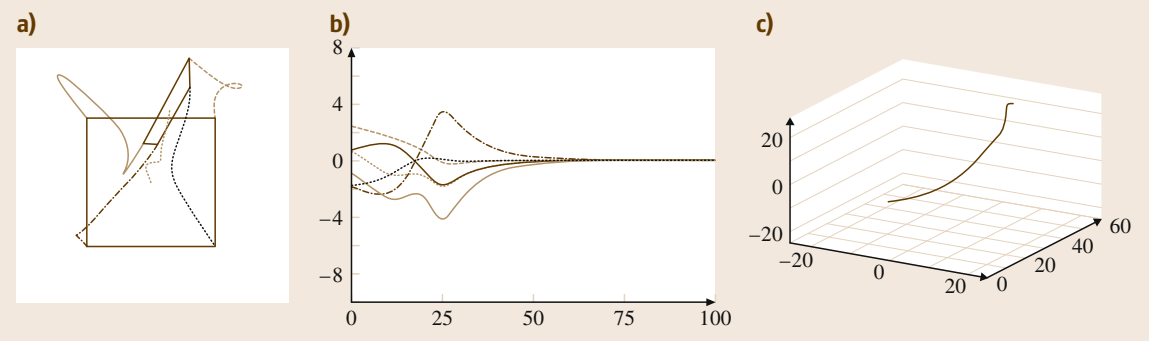


Fig.34.2a–c IBVS system behavior using $\boldsymbol{s} = (x_1, y_1, \ldots, x_4, y_4)$ and $\widehat{\mathbf{L}_e^+} = \mathbf{L}_{e^*}^+$: **(a)** image point trajectories including the trajectory of the center of the square, which is not used in the control scheme, **(b)** $\boldsymbol{v}_c$ components (cm/s and deg/s) computed at each iteration of the control scheme, and **(c)** the 3-D trajectory of the camera optical center expressed in the desired camera frame $\mathcal{R}_{c^*}$ (cm)

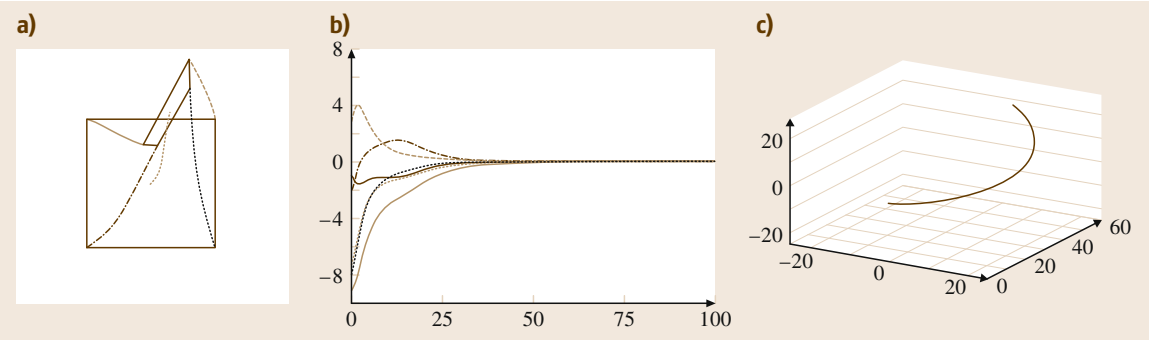


Fig.34.3a–c IBVS system behavior using $\boldsymbol{s} = (x_1, y_1, \ldots, x_4, y_4)$ and $\widehat{\mathbf{L}_e^+} = \mathbf{L}_e^+$ (refer to Fig. 34.2a-c for description)

that the initial camera pose has been selected far away from the desired pose, particularly with regard to the rotational motions, which are known to be the most problematic for IBVS. In the simulations presented in the following, no noise or modeling errors have been introduced in order to allow comparison of different behaviors in perfect conditions. The accompanying videos show experimental results obtained using an Adept Viper robot arm and the ViSP library [34.11]. The videos all show the same task being performed from the same initial conditions, and only the control approach varies.

The results obtained by using

$$\widehat{\mathbf{L}_e^+} = \mathbf{L}_{e^*}^+$$

are given in Fig. 34.2 and VIDEO 59. Note that despite the large displacement that is required the system converges. However, neither the behavior in the image, nor the computed camera velocity components, nor the 3-D trajectory of the camera present desirable properties far from the convergence (i. e., for the first 30 or so iterations).

The results obtained using

$$\widehat{\mathbf{L}_e^+} = \mathbf{L}_e^+$$

are given in Fig. 34.3 and VIDEO 60. In this case, the trajectories of the points in the image are almost straight lines, which means that the points never leave the camera's field of view. However the behavior induced in the camera frame is even less satisfactory than for the case of

$$\widehat{\mathbf{L}_e^+} = \mathbf{L}_{e^*}^+ .$$

The large camera velocities at the beginning of the servo indicate that the condition number of $\widehat{\mathbf{L}_e^+}$ is high at the start of the trajectory, and the camera trajectory is far from a straight line.

The choice

$$\widehat{\mathbf{L}_e^+} = \left(\frac{\mathbf{L}_e}{2} + \frac{\mathbf{L}_{e^*}}{2}\right)^+$$

provides good performance in practice. Indeed, as can be seen in Fig. 34.4, the camera velocity components do not include large oscillations, which provides

Avidbots Ex1016
Page 879 of 2154

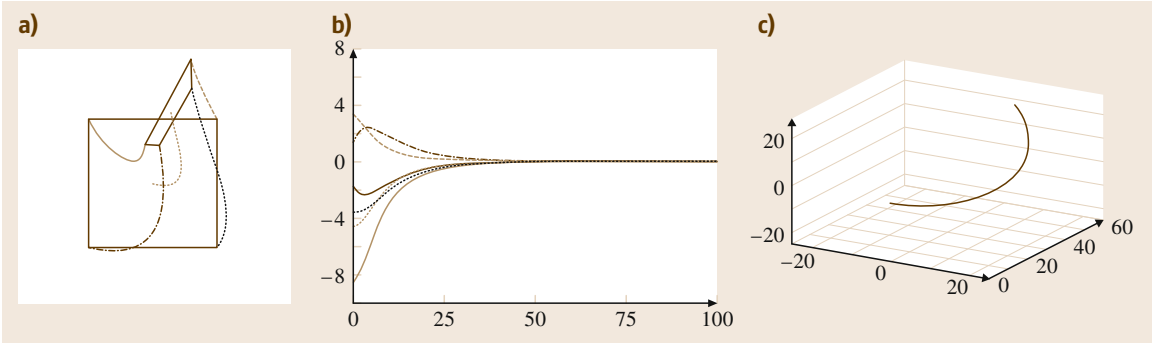


Fig. 34.4 IBVS system behavior using $\boldsymbol{s} = (x_1, y_1, \ldots, x_4, y_4)$ and $\widehat{\mathbf{L}_e^+} = \left(\mathbf{L}_e/2 + \mathbf{L}_e^*/2\right)^+$ (refer to Fig. 34.2a-c for description)

a smooth trajectory in both the image and in 3-D space (VIDEO 61).

34.2.3 A Geometrical Interpretation of IBVS

It is quite easy to provide a geometric interpretation of the behavior of the control schemes defined above. The example illustrated in Fig. 34.5 corresponds to a pure rotation around the optical axis from the initial configuration (shown in blue) to the desired configuration of four coplanar points parallel to the image plane (shown in red).

As explained above, using $\mathbf{L}_e^+$ in the control scheme attempts to ensure an exponential decrease of the error $\boldsymbol{e}$. This means that, when x and y image point coordinates compose this error, the points' trajectories in the image follow straight lines from their initial to their desired positions, when this is possible. This leads to the image motion plotted in green in the figure. The camera motion to realize this image motion can be easily deduced and is indeed composed of a rotational motion around the optical axis, but combined with a retreating translational motion along the optical axis [34.12]. This unexpected motion is due to the choice of the features and the form of the third and sixth columns in the interaction matrix, which induces a coupling between the features and the two degrees of freedom involved (v_z and ω_z). If the rotation between the initial and desired configurations is very large, this phenomenon is amplified, and leads to a particular case for a rotation of π radians where no rotational motion at all will be induced by the control scheme [34.13]. On the other hand, when the rotation is small, this phenomenon almost disappears. To conclude, the behavior is locally satisfactory (i. e., when the error is small), but it can be unsatisfactory when the error is large. As we will see below, these results are consistent with the local asymptotic stability results that can be obtained for IBVS.

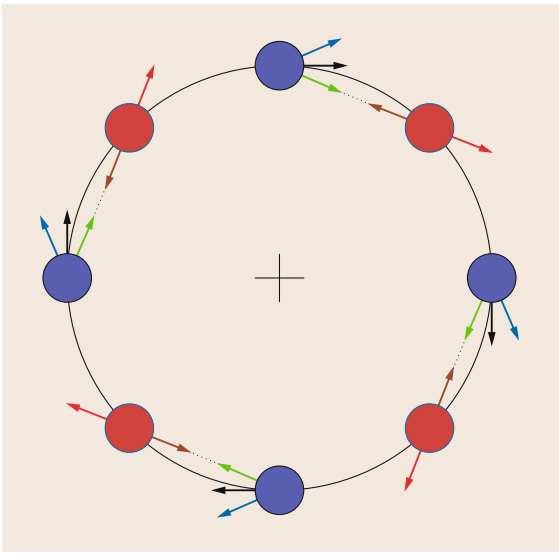


Fig. 34.5 Geometrical interpretation of IBVS: going from the *blue position* to the *red one*. In *green*, image motion when $\mathbf{L}_e^+$ is used in the control scheme; in *blue*, when $\mathbf{L}_{e^*}^+$ is used; and in *black* when $(\mathbf{L}_e/2 + \mathbf{L}_{e^*}/2)^+$ is used (see the text for more details)

If instead we use $\mathbf{L}_{e^*}^+$ in the control scheme, the image motion generated can easily be shown to be the blue one plotted in Fig. 34.5. Indeed, if we consider the same control scheme as before but starting from $\boldsymbol{s}^*$ to reach $\boldsymbol{s}$, we obtain

$$\boldsymbol{v}_c = -\lambda \mathbf{L}_{e^*}^+ (\boldsymbol{s}^* - \boldsymbol{s}) \,,$$

which again induces straight-line trajectories from the red points to the blue ones, causing the image motion plotted in brown. Going back to our problem, the control scheme computes a camera velocity that is exactly the opposite one

$$\boldsymbol{v}_c = -\lambda \mathbf{L}_{e^*}^+ (\boldsymbol{s} - \boldsymbol{s}^*) \,,$$

and thus generates the image motion plotted in red at the red points. Transformed to the blue points, the camera velocity generates the blue image motion and corresponds once again to a rotational motion around the optical axis, combined now with an unexpected forward motion along the optical axis. The same analysis can be done as before for the case of large or small errors. We can add that, as soon as the error decreases significantly, both control schemes get closer, and tend to the same one (since $\mathbf{L}_e = \mathbf{L}_{e^*}$ when $\boldsymbol{e} = \boldsymbol{e}^*$) with a nice behavior characterized with the image motion plotted in black and a camera motion composed of only a rotation around the optical axis when the error tends towards zero.

If we instead use

$$\widehat{\mathbf{L}_e^+} = \left(\frac{\mathbf{L}_e}{2} + \frac{\mathbf{L}_{e^*}}{2}\right)^+ ,$$

it is intuitively clear that considering the mean of $\mathbf{L}_e$ and $\mathbf{L}_{e^*}$ generates the image motion plotted in black, even when the error is large. In all cases but the rotation of π radians, the camera motion is now a pure rotation around the optical axis, without any unexpected translational motion.

34.2.4 Stability Analysis

We now consider the fundamental issues related to the stability of IBVS. To assess the stability of the closed-loop visual servo systems, we will use Lyapunov analysis. In particular, consider the candidate Lyapunov function defined by the squared error norm $\mathcal{L} = \frac{1}{2}\|\boldsymbol{e}(t)\|^2$, whose derivative is given by

$$\dot{\mathcal{L}} = \boldsymbol{e}^\mathrm{T}\dot{\boldsymbol{e}} = -\lambda \boldsymbol{e}^\mathrm{T}\mathbf{L}_e\widehat{\mathbf{L}_e^+}\boldsymbol{e}$$

since $\dot{\boldsymbol{e}}$ is given by (34.6). The global asymptotic stability of the system is thus obtained when the following sufficient condition is satisfied

$$\mathbf{L}_e\widehat{\mathbf{L}_e^+} > 0\,. \tag{34.13}$$

If the number of features is equal to the number of camera degrees of freedom (i. e., $k = 6$), and if the features are chosen and the control scheme designed so that $\mathbf{L}_e$ and $\widehat{\mathbf{L}_e^+}$ are of full rank 6, then the condition (34.13) is satisfied if the approximations involved in $\widehat{\mathbf{L}_e^+}$ are not too coarse.

As discussed above, for most IBVS approaches we have $k > 6$. Therefore the condition (34.13) can never be ensured since

$$\mathbf{L}_e\widehat{\mathbf{L}_e^+} \in \mathbb{R}^{k\times k}$$

is at most of rank 6, and thus $\mathbf{L}_e\widehat{\mathbf{L}_e^+}$ has a nontrivial null space. In this case, configurations such that $\boldsymbol{e} \in \ker \widehat{\mathbf{L}_e^+}$ correspond to local minima. Reaching such a local minimum is illustrated in Fig. 34.6. As can be seen in Fig. 34.6d, each component of $\boldsymbol{e}$ has a nice exponential decrease with the same convergence speed, causing straight-line trajectories to be realized in the image, but the error reached is not exactly zero, and it is clear from Fig. 34.6c that the system has been attracted to a local minimum far away from the desired configuration. Thus, only local asymptotic stability can be obtained for IBVS.

To study local asymptotic stability when $k > 6$, let us first define a new error $\boldsymbol{e}'$ with $\boldsymbol{e}' = \widehat{\mathbf{L}_e^+}\boldsymbol{e}$. The time derivative of this error is given by

$$\dot{\boldsymbol{e}}' = \widehat{\mathbf{L}_e^+}\dot{\boldsymbol{e}} + \dot{\widehat{\mathbf{L}_e^+}}\boldsymbol{e} = \left(\widehat{\mathbf{L}_e^+}\mathbf{L}_e + \boldsymbol{O}\right)\boldsymbol{v}_\mathrm{c}\,,$$

where $\boldsymbol{O} \in \mathbb{R}^{6\times 6}$ is equal to $\mathbf{0}$ when $\boldsymbol{e} = 0$, whatever the choice of $\widehat{\mathbf{L}_e^+}$ [34.14]. Using the control scheme (34.5),w e obtain

$$\dot{\boldsymbol{e}}' = -\lambda\left(\widehat{\mathbf{L}_e^+}\mathbf{L}_e + \boldsymbol{O}\right)\boldsymbol{e}'\,,$$

which is known to be locally asymptotically stable in a neighborhood of $\boldsymbol{e} = \boldsymbol{e}^* = \mathbf{0}$ if

$$\widehat{\mathbf{L}_e^+}\mathbf{L}_e > 0\,, \tag{34.14}$$

where $\widehat{\mathbf{L}_e^+}\mathbf{L}_e \in \mathbb{R}^{6\times 6}$. Indeed, only the linearized system

$$\dot{\boldsymbol{e}}' = -\lambda\widehat{\mathbf{L}_e^+}\mathbf{L}_e\boldsymbol{e}'$$

has to be considered if we are interested in the local asymptotic stability [34.15].

Once again, if the features are chosen and the control scheme designed so that $\mathbf{L}_e$ and $\widehat{\mathbf{L}_e^+}$ are of full rank 6, then condition (34.14) is ensured if the approximations involved in $\widehat{\mathbf{L}_e^+}$ are not too coarse.

To end the demonstration of local asymptotic stability, we must show that there does not exist any configuration $\boldsymbol{e} \neq \boldsymbol{e}^*$ such that $\boldsymbol{e} \in \ker \widehat{\mathbf{L}_e^+}$ in a small neighborhood of $\boldsymbol{e}^*$ and in a small neighborhood of the corresponding pose $\boldsymbol{p}^*$. Such configurations correspond to local minima where $\boldsymbol{v}_\mathrm{c} = 0$ and $\boldsymbol{e} \neq \boldsymbol{e}^*$. If such a pose $\boldsymbol{p}$ would exist, it is possible to restrict the neighborhood around $\boldsymbol{p}^*$ so that there exists a camera velocity $\boldsymbol{v}$ to reach $\boldsymbol{p}^*$ from $\boldsymbol{p}$. This camera velocity would imply a variation of the error $\dot{\boldsymbol{e}} = \mathbf{L}_e\boldsymbol{v}$. However, such a variation cannot belong to $\ker \widehat{\mathbf{L}_e^+}$ since $\widehat{\mathbf{L}_e^+}\mathbf{L}_e > 0$. Therefore, we have $\boldsymbol{v}_\mathrm{c} = 0$ if and only if $\dot{\boldsymbol{e}} = \mathbf{0}$, i. e., $\boldsymbol{e} = \boldsymbol{e}^*$, in a neighborhood of $\boldsymbol{p}^*$.

Avidbots Ex1016
Page 881 of 2154

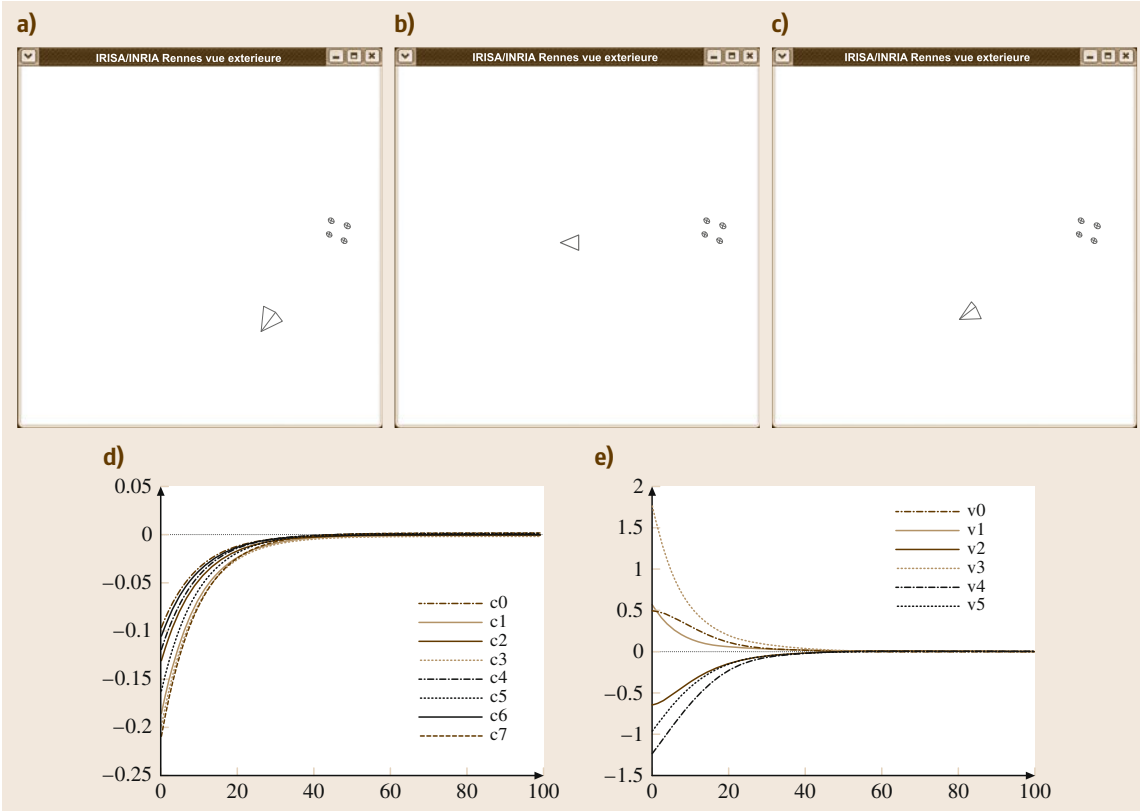


Fig.34.6a–e IBVS reaching a local minimum using $\boldsymbol{s} = (x_1, y_1, \ldots, x_4, y_4)$ and $\widehat{\mathbf{L}_{\boldsymbol{e}}^{+}} = \mathbf{L}_{\boldsymbol{e}}^{+}$: **(a)** the initial configuration, **(b)** the desired one, **(c)** the configuration reached after the convergence of the control scheme, **(d)** the evolution of the error $\boldsymbol{e}$ at each iteration of the control scheme, and **(e)** the evolution of the six components of the camera velocity $\boldsymbol{v}_c$

Even though local asymptotic stability can be ensured when $k > 6$, we recall that global asymptotic stability cannot be ensured. For instance, as illustrated in Fig. 34.6, there may exist local minima corresponding to configurations where $\boldsymbol{e} \in \ker \widehat{\mathbf{L}_{\boldsymbol{e}}^{+}}$, which are outside of the neighborhood considered above. Determining the size of the neighborhood in which stability and the convergence are ensured is still an open issue, even if this neighborhood is surprisingly quite large in practice.

34.2.5 IBVS with a Stereo Vision System

It is straightforward to extend the IBVS approach to a multicamera system. If a stereo vision system is used, and a world point is visible in both left and right images (Fig. 34.7), it is possible to use as visual features

$$\boldsymbol{s} = \boldsymbol{x}_s = (\boldsymbol{x}_l, \boldsymbol{x}_r) = (x_l, y_l, x_r, y_r)$$

i. e., to represent the point by just stacking in $\boldsymbol{s}$ the x and y coordinates of the observed point in the left and right images [34.16]. However, care must be taken when constructing the corresponding interaction matrix since the form given in (34.11) is expressed in either the left or right camera frame. More precisely, we have

$$\begin{cases} \dot{\boldsymbol{x}}_l &= \mathbf{L}_{\boldsymbol{x}_l}\, \boldsymbol{v}_l \,, \\ \dot{\boldsymbol{x}}_r &= \mathbf{L}_{\boldsymbol{x}_r}\, \boldsymbol{v}_r \,, \end{cases}$$

where $\boldsymbol{v}_l$ and $\boldsymbol{v}_r$ are the spatial velocity of the left and right camera, respectively, and where the analytical form of $\mathbf{L}_{\boldsymbol{x}_l}$ and $\mathbf{L}_{\boldsymbol{x}_r}$ are given by (34.12).

By choosing a sensor frame rigidly linked to the stereo vision system, we obtain

$$\dot{\boldsymbol{x}}_s = \begin{pmatrix} \dot{\boldsymbol{x}}_l \\ \dot{\boldsymbol{x}}_r \end{pmatrix} = \mathbf{L}_{\boldsymbol{x}_s}\, \boldsymbol{v}_s \,,$$

where the interaction matrix related to $\boldsymbol{x}_s$ can be determined using the spatial motion transform matrix $\mathbf{V}$ defined in Chap. 02 to transform velocities expressed in the left or right cameras frames to the sensor frame. We

Avidbots Ex1016
Page 882 of 2154

recall that $\mathbf{V}$ is given by

$$\mathbf{V} = \begin{pmatrix} \mathbf{R} & [\boldsymbol{t}]_\times \mathbf{R} \\ \mathbf{0} & \mathbf{R} \end{pmatrix} , \tag{34.15}$$

where $[\boldsymbol{t}]_\times$ is the skew-symmetric matrix associated to the vector $\boldsymbol{t}$ and where $(\mathbf{R}, \boldsymbol{t}) \in SE(3)$ is the rigid-body transformation from the camera to the sensor frame. The numerical values for these matrices are directly obtained from the calibration step of the stereo vision system. Using this equation, we obtain

$$\mathbf{L}_{x_s} = \begin{pmatrix} \mathbf{L}_{x_l}\, {}^{l}\mathbf{V}_s \\ \mathbf{L}_{x_r}\, {}^{r}\mathbf{V}_s \end{pmatrix} .$$

Note that $\mathbf{L}_{x_s} \in \mathbb{R}^{4\times 6}$ is always of rank 3 because of the epipolar constraint that links the perspective projection of a 3-D point in a stereo vision system (Fig. 34.7). Another simple interpretation is that a 3-D point is represented by three independent parameters, which makes it impossible to find more than three independent parameters using any sensor observing that point.

To control the six degrees of freedom of the system, it is necessary to consider at least three points, as the rank of the interaction matrix considering only two points is 5.

Using a stereo vision system, since the 3-D coordinates of any point observed in both images can be easily estimated by a simple triangulation process it is possible and quite natural to use these 3-D coordinates in the features set $\boldsymbol{s}$. Such an approach would be, strictly speaking, a position-based approach, since it would require 3-D parameters in $\boldsymbol{s}$.

34.2.6 IBVS with Cylindrical Coordinates of Image Points

In the previous sections, we have considered the Cartesian coordinates of image points. As proposed in [34.17] it may be useful to consider instead the cylindrical coordinates $\boldsymbol{\gamma} = (\rho, \theta)$ of the image points instead of their Cartesian coordinates $\boldsymbol{x} = (x, y)$. They are given by

$$\rho = \sqrt{x^2 + y^2} \ , \ \theta = \arctan \frac{y}{x}$$

from which we deduce

$$\dot{\rho} = \frac{x\dot{x} + y\dot{y}}{\rho} \ , \ \dot{\theta} = \frac{x\dot{y} - y\dot{x}}{\rho^2} \ .$$

Using (34.11) and then substituting x by $\rho\cos\theta$ and y by $\rho\sin\theta$, we obtain immediately

$$\mathbf{L}_{\boldsymbol{\gamma}} = \begin{pmatrix} \frac{-c}{Z} & \frac{-s}{Z} & \frac{\rho}{Z} & (1+\rho^2)s & -(1+\rho^2)c & 0 \\ \frac{s}{\rho Z} & \frac{-c}{\rho Z} & 0 & \frac{c}{\rho} & \frac{s}{\rho} & -1 \end{pmatrix} , \tag{34.16}$$

where $c = \cos\theta$ and $s = \sin\theta$. Note that θ is not defined when the image point lies at the principal point (where $x = y = \rho = 0$). It is thus not surprising that the interaction matrix $\mathbf{L}_{\boldsymbol{\gamma}}$ is singular in that case.

If we go back to the example depicted in Fig. 34.5, the behavior obtained using cylindrical coordinates will be the expected one, that is a pure rotation around the optical axis, by using either $\mathbf{L}_{e}^{+}$, $\mathbf{L}_{e^*}^{+}$ or $(\mathbf{L}_e/2 + \mathbf{L}_{e^*}/2)^{+}$ in the control scheme. This is due to the form of the third and sixth columns of the interaction matrix (34.16), which leads to a decoupled system.

34.2.7 IBVS with Other Geometrical Features

In the previous sections, we have only considered image point coordinates in $\boldsymbol{s}$. Other geometrical primitives can of course be used. There are several reasons to do so. Firstly, the scene observed by the camera cannot always be described merely by a collection of points, in which case the image processing provides other types of measurements, such as a set of straight lines or the contours of an object. Secondly, richer geometric primitives may ameliorate the decoupling and linearizing issues that motivate the design of partitioned systems (Sect. 34.4). Finally, the robotic task to be achieved may be expressed in terms of virtual linkages (or fixtures) between the camera and the observed objects [34.18, 19], sometimes expressed directly by constraints between primitives, such as point-to-line [34.20] (which means that an observed point must lie on a specified line).

It is possible to determine the interaction matrix related to the perspective projection of a large class of geometrical primitives, such as segments, straight lines, spheres, circles, and cylinders. The results are given in [34.3, 18]. Recently, the analytical form of the interaction matrix related to any image moments corresponding to planar objects has been computed. This

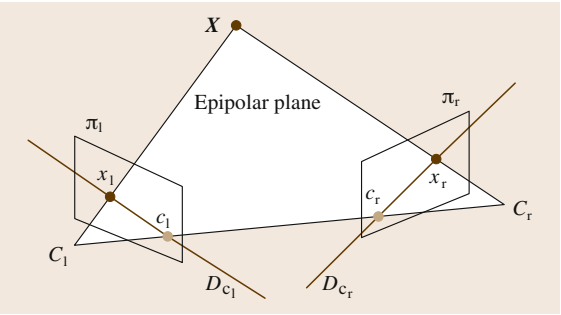


Fig. 34.7 A stereo vision system

Avidbots Ex1016
Page 883 of 2154

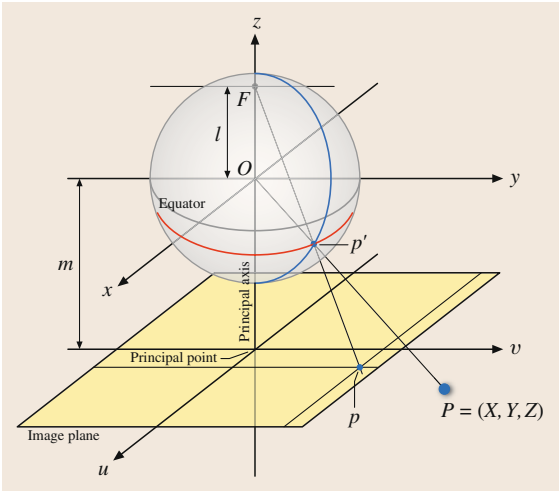


Fig. 34.8 Unified imaging model of Geyer and Daniilidis (after [34.7])

makes it possible to consider planar objects of any shape [34.21]. If a collection of points is measured in the image, moments can also be used [34.22]. In both cases, moments allow the use of intuitive geometrical features, such as the center of gravity or the orientation of an object. By selecting an adequate combination of moments, it is then possible to determine partitioned systems with good decoupling and linearizing properties [34.21, 22].

Note that, for all these features (geometrical primitives, moments), the depth of the primitive or of the object considered appears in the coefficients of the interaction matrix related to the translational degrees of freedom, as was the case for the image points. An estimation of this depth is thus generally necessary (Sect. 34.6). In a few situations, for instance with a suitable normalization of moments [34.22], only the constant desired depth appears in the interaction matrix, which makes estimating depth unnecessary.

34.2.8 Non-Perspective Cameras

The vast majority of cameras we use, and our own eyes, are characterized by a perspective projection model which closely approximate the ideal pinhole imaging model. Such cameras have a narrow field of view, typically less than one half hemisphere. For robotics it is often advantageous to have a large field of view and this can be achieved using a fisheye lens camera or a catadioptric (lens and mirror system) camera (often referred to as a panoramic camera). For these non-perspective sensors the interaction matrix of any visual feature has a different form to those discussed above, such as (34.12) and (34.16) for an image point.

Rather than determine the interaction matrix of visual features expressed in the image plane of non-perspective cameras, we can transform the images from these cameras to the view that would be seen by an ideal spherical camera. The spherical model projects world points onto a unit sphere, the intersection of the unit sphere with the ray from the world point to the center of the sphere. Such an ideal camera has the largest possible field of view. The unified imaging model [34.23] shown in Fig 34.8b provides a general mechanism to project a world point to the image plane of a large class of cameras. To be precise it includes all central projection cameras and this includes perspective and some catadioptric cameras with particular mirror shapes, but in practice it is a very good approximation to non-central cameras including fisheye and general catadioptric systems. The mechanism of this unified model can also be used to reproject points from these varied image planes to a spherical camera.

Referring to Fig. 34.8 the world point **P** is represented by the vector $\boldsymbol{X} = (X, Y, Z)$ in the camera frame, and is projected onto the surface of the unit sphere at the point $\boldsymbol{x}_s = (x_s, y_s, z_s)$ with

$$x_s = \frac{X}{R}\,, \quad y_s = \frac{Y}{R}\,, \quad \text{and} \quad z_s = \frac{Z}{R}\,,$$

where $R = \sqrt{X^2 + Y^2 + Z^2}$ is the distance from the sphere center to the world point.

The interaction matrix of $\boldsymbol{x}_s$ is derived in essentially the same manner as for the perspective camera, and it can be shown to be [34.24]

$$\mathbf{L}_{\boldsymbol{x}_s} = \left[\frac{1}{R}\left(\boldsymbol{x}_s \boldsymbol{x}_s{}^{\mathrm{T}} - \mathbf{I}_3\right)[\boldsymbol{x}_s]_{\times}\right]\,. \tag{34.17}$$

Note that R can be expressed as a function of the point depth Z by using $R = Z\sqrt{1 + x_s^2 + y_s^2}$. Therefore, the general spherical model does not add any supplementary unknown in the interaction matrix.

A particular advantage of the spherical model is that for pure camera rotation the shape of an object is invariant, which eases determining visual features that are only linked to translational motions.

34.2.9 Direct Estimation

In the previous sections, we have focused on the analytical form of the interaction matrix. It is also possible to estimate its numerical value directly using either an offline learning step, or an online estimation scheme.

All the methods proposed to estimate the interaction matrix numerically rely on the observation of a variation of the features due to a known or measured camera motion. More precisely, if we measure a feature's variation $\Delta \boldsymbol{s}$ due to a camera motion $\Delta \boldsymbol{v}_c$, we

have from (34.2)

$$\mathbf{L}_s \Delta \boldsymbol{v}_c = \Delta \boldsymbol{s} \,,$$

which provides k equations while we have $k \times 6$ unknown values in $\mathbf{L}_s$. Using a set of N independent camera motions with $N > 6$, it is thus possible to estimate $\mathbf{L}_s$ by solving

$$\mathbf{L}_s \boldsymbol{A} = \boldsymbol{B} \,,$$

where the columns of $\boldsymbol{A} \in \mathbb{R}^{6\times N}$ and $\boldsymbol{B} \in \mathbb{R}^{k\times N}$ are, respectively, formed from the set of camera motions and the set of corresponding features variations. The least-square solution is of course given by

$$\widehat{\mathbf{L}_s} = \boldsymbol{B}\boldsymbol{A}^{+} \,. \tag{34.18}$$

Methods based on neural networks have also been developed to estimate $\mathbf{L}_s$ [34.25, 26]. It is also possible to estimate the numerical value of $\mathbf{L}_s^{+}$ directly, which in practice provides a better behavior [34.27]. In this case, the basic relation is

$$\mathbf{L}_s^{+} \Delta \boldsymbol{s} = \Delta \boldsymbol{v}_c \,,$$

which provides six equations. Using a set of N measurements, with $N > k$, we now obtain

$$\widehat{\mathbf{L}_s^{+}} = \boldsymbol{A}\boldsymbol{B}^{+} \,. \tag{34.19}$$

In the first case (34.18), the six columns of $\mathbf{L}_s$ are estimated by solving six linear systems, while in the second case (34.19), the k columns of $\mathbf{L}_s^{+}$ are estimated by solving k linear systems, which explains the difference in the results.

Estimating the interaction matrix online can be viewed as an optimization problem, and consequently a number of researchers have investigated approaches that derive from optimization methods. These methods typically discretize the system equation (34.2), and use an iterative updating scheme to refine the estimate of $\widehat{\mathbf{L}_s}$ at each stage. One such online and iterative formulation uses the Broyden update rule given by [34.28, 29]

$$\widehat{\mathbf{L}_s}(t+1) = \widehat{\mathbf{L}_s}(t) + \frac{\alpha}{\Delta \boldsymbol{v}_c^{\mathrm{T}} \Delta \boldsymbol{v}_c} \left[\Delta \boldsymbol{x} - \widehat{\mathbf{L}_s}(t) \Delta \boldsymbol{v}_c \right] \Delta \boldsymbol{v}_c^{\mathrm{T}} \,,$$

where α defines the update speed. This method has been generalized to the case of moving objects in [34.30].

The main interest of using such numerical estimations in the control scheme is that it avoids all the modeling and calibration steps. It is particularly useful when using features whose interaction matrix is not available in analytical form. For instance, in [34.31], the main eigenvalues of the principal component analysis of an image have been considered in a visual servoing scheme. The drawback of these methods is that no theoretical stability and robustness analysis can be made.

34.3 Pose-Based Visual Servo

Pose-based control schemes (PBVS) [34.1, 32, 33] use the pose of the camera with respect to some reference coordinate frame to define $\boldsymbol{s}$. Computing this pose from a set of measurements in one image necessitates the camera intrinsic parameters and the 3-D model of the object observed to be known. This classic computer vision problem is called the 3-D localization problem. While this problem is beyond the scope of the present chapter, many solutions have been presented in the literature [34.34, 35] and its basic principles are recalled in Chap. 32.

It is then typical to define $\boldsymbol{s}$ in terms of the parameterization used to represent the camera pose. Note that the parameters $\boldsymbol{a}$ involved in the definition (34.1) of $\boldsymbol{s}$ are now the camera intrinsic parameters and the 3-D model of the object.

It is convenient to consider three coordinate frames: the current camera frame $\mathcal{F}_c$, the desired camera frame $\mathcal{F}_{c^*}$, and a reference frame $\mathcal{F}_o$ attached to the object. We adopt here the standard notation of using a leading superscript to denote the frame with respect to which a set of coordinates is defined. Thus, the coordinate vectors ${}^{c}\boldsymbol{t}_o$ and ${}^{c^*}\boldsymbol{t}_o$ give the coordinates of the origin of the object frame expressed relative to the current camera frame, and relative to the desired camera frame, respectively. Furthermore, let $\mathbf{R} = {}^{c^*}\mathbf{R}_c$ be the rotation matrix that gives the orientation of the current camera frame relative to the desired frame.

We can define $\boldsymbol{s}$ to be $(\boldsymbol{t}, \theta\boldsymbol{u})$, in which $\boldsymbol{t}$ is a translation vector, and $\theta\boldsymbol{u}$ gives the angle/axis parameterization for the rotation. We now discuss two choices for $\boldsymbol{t}$, and give the corresponding control laws.

If $\boldsymbol{t}$ is defined relative to the object frame $\mathcal{F}_o$, we obtain

$$\boldsymbol{s} = ({}^{c}\boldsymbol{t}_o, \theta\boldsymbol{u}) \,, \boldsymbol{s}^* = ({}^{c^*}\boldsymbol{t}_o, 0) \,,$$

and

$$\boldsymbol{e} = ({}^{c}\boldsymbol{t}_o - {}^{c^*}\boldsymbol{t}_o, \theta\boldsymbol{u}) \,.$$

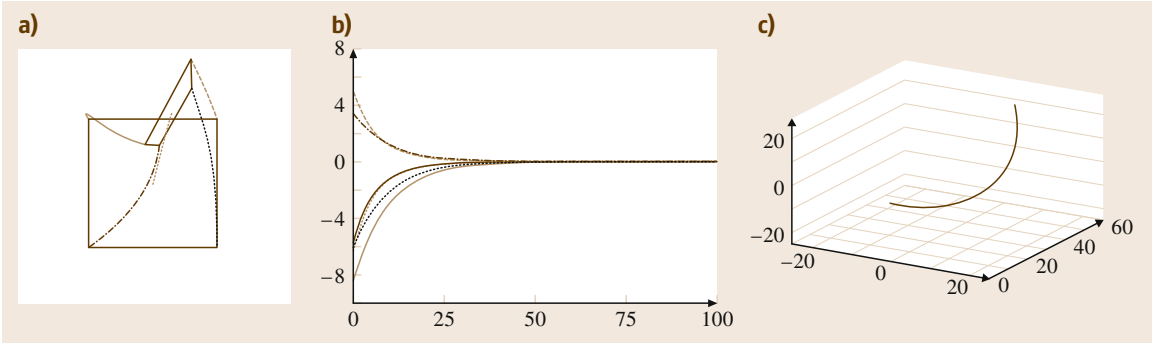


Fig. 34.9 PBVS system behavior using $\boldsymbol{s} = ({}^{c}\boldsymbol{t}_{o}, \theta\boldsymbol{u})$ (refer to Fig. 34.2a-c for description)

In this case, the interaction matrix related to $\boldsymbol{e}$ is given by

$$\mathbf{L}_{\boldsymbol{e}} = \begin{pmatrix} -\mathbf{I}_3 & [{}^{c}\boldsymbol{t}_{o}]_{\times} \\ \mathbf{0} & \mathbf{L}_{\theta\boldsymbol{u}} \end{pmatrix} , \tag{34.20}$$

in which $\mathbf{I}_3$ is the 3×3 identity matrix and $\mathbf{L}_{\theta\boldsymbol{u}}$ is given by [34.36]

$$\mathbf{L}_{\theta\boldsymbol{u}} = \mathbf{I}_3 + \frac{\theta}{2}[\boldsymbol{u}]_{\times} + \left(1 - \frac{\operatorname{sinc}\theta}{\operatorname{sinc}^2\theta/2}\right)[\boldsymbol{u}]_{\times}^2 , \tag{34.21}$$

where $\operatorname{sinc} x$ is the sinus cardinal defined such that $x \operatorname{sinc} x = \sin x$ and $\operatorname{sinc} 0 = 1$.

Following the development in Sect. 34.1, we obtain the control scheme

$$\boldsymbol{v}_{c} = -\lambda \widehat{\mathbf{L}_{\boldsymbol{e}}^{-1}} \boldsymbol{e}$$

since the dimension k of $\boldsymbol{s}$ is six, that is, the number of camera degrees of freedom. By setting

$$\widehat{\mathbf{L}_{\boldsymbol{e}}^{-1}} = \begin{pmatrix} -\mathbf{I}_3 & [{}^{c}\boldsymbol{t}_{o}]_{\times}\mathbf{L}_{\theta\boldsymbol{u}}^{-1} \\ \mathbf{0} & \mathbf{L}_{\theta\boldsymbol{u}}^{-1} \end{pmatrix} , \tag{34.22}$$

we obtain after simple developments

$$\begin{cases} \boldsymbol{v}_{c} = -\lambda\left[{}^{c^*}\boldsymbol{t}_{o} - {}^{c}\boldsymbol{t}_{o}\right] + [{}^{c}\boldsymbol{t}_{o}]_{\times}\,\theta\boldsymbol{u} \\ \boldsymbol{\omega}_{c} = -\lambda\theta\boldsymbol{u} \end{cases} . \tag{34.23}$$

since $\mathbf{L}_{\theta\boldsymbol{u}}$ is such that

$$\mathbf{L}_{\theta\boldsymbol{u}}^{-1}\theta\boldsymbol{u} = \theta\boldsymbol{u} .$$

Ideally, that is, if the pose parameters are perfectly estimated, the behavior of $\boldsymbol{e}$ will be the expected one ($\dot{\boldsymbol{e}} = -\lambda\boldsymbol{e}$). The choice of $\boldsymbol{e}$ causes the rotational motion to follow a geodesic with an exponential decreasing speed and causes the translational parameters involved in $\boldsymbol{s}$ to decrease at the same speed. This explains the nice exponential decrease of the camera velocity components in Fig. 34.9. Furthermore, the trajectory in the image of the origin of the object frame follows a straight line (here the center of the four points has been selected as this origin). On the other hand, the camera trajectory does not follow a straight line (VIDEO 62).

Another PBVS scheme can be designed by using $\boldsymbol{s} = ({}^{c^*}\boldsymbol{t}_{c}, \theta\boldsymbol{u})$. In this case, we have $\boldsymbol{s}^* = \mathbf{0}$, $\boldsymbol{e} = \boldsymbol{s}$, and

$$\mathbf{L}_{\boldsymbol{e}} = \begin{pmatrix} \mathbf{R} & \mathbf{0} \\ \mathbf{0} & \mathbf{L}_{\theta\boldsymbol{u}} \end{pmatrix} . \tag{34.24}$$

Note the decoupling between translational and rotational motions, which allows us to obtain a simple control scheme

$$\begin{cases} \boldsymbol{v}_{c} &= -\lambda\mathbf{R}^{\mathrm{T}}\,{}^{c^*}\boldsymbol{t}_{c} \\ \boldsymbol{\omega}_{c} &= -\lambda\,\theta\boldsymbol{u} \end{cases} . \tag{34.25}$$

In this case, as can be seen in Fig. 34.10 and in VIDEO 63, if the pose parameters involved in (34.25) are estimated perfectly, the camera trajectory is a straight line, while the image trajectories are less satisfactory than before. Some particular configurations can be found which will lead to some points leaving the camera field of view during the robot motion.

The stability properties of PBVS seem quite attractive. Since $\mathbf{L}_{\theta\boldsymbol{u}}$ given in (34.21) is nonsingular when $\theta \neq 2k\pi$, $\forall k \in Z^*$, we obtain from (34.13) the global asymptotic stability of the system since $\mathbf{L}_{\boldsymbol{e}}\widehat{\mathbf{L}_{\boldsymbol{e}}^{-1}} = \mathbf{I}_6$, under the strong hypothesis that all the pose parameters are perfect. This is true for both methods presented above, since the interaction matrices given in (34.20) and (34.24) are full rank when $\mathbf{L}_{\theta\boldsymbol{u}}$ is nonsingular.

Avidbots Ex1016
Page 886 of 2154

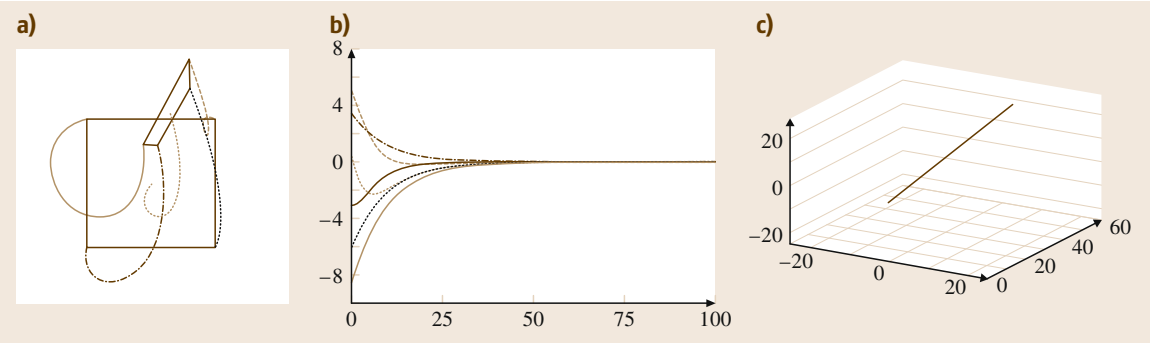


Fig. 34.10 PBVS system behavior using $\boldsymbol{s} = ({}^{c^*}\boldsymbol{t}_c, \theta\boldsymbol{u})$ (refer to Fig. 34.2a-c for description)

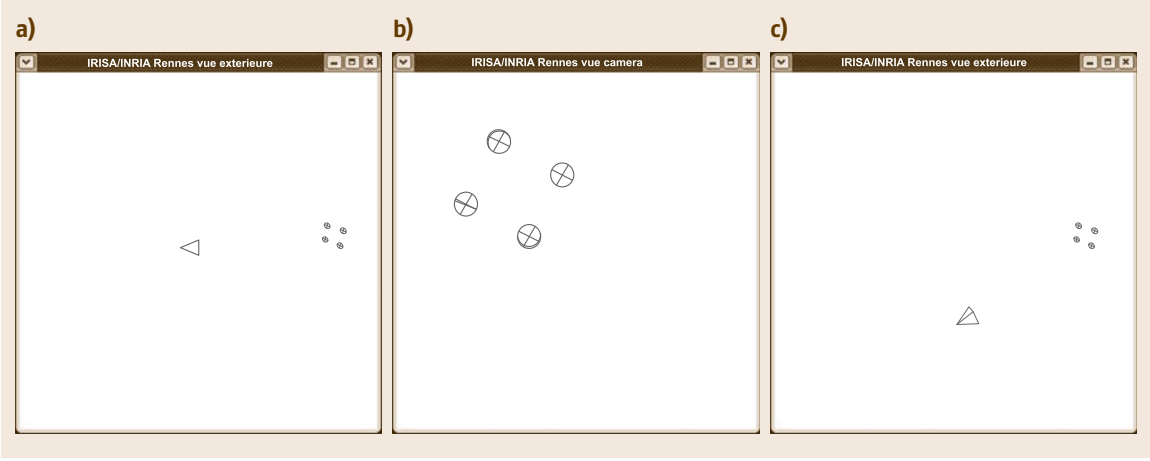


Fig.34.11a–c Two different camera poses **(a,c)** that provide almost the same image of four coplanar points shown overlaid in **(b)**

With regard to robustness, feedback is computed using *estimated* quantities that are a function of the image measurements and the system calibration parameters. For the first method presented in Sect. 34.3 (the analysis for the second method is analogous), the interaction matrix given in (34.20) corresponds to perfectly estimated pose parameters, while the real one is unknown since the estimated pose parameters may be biased due to calibration errors, or inaccurate and unstable due to noise [34.13]. The true positivity condition (34.13) should in fact be written

$$\mathbf{L}_{\hat{e}}\widehat{\mathbf{L}_{\hat{e}}^{-1}} > 0\,, \tag{34.26}$$

where $\widehat{\mathbf{L}_{\hat{e}}^{-1}}$ is given by (34.22) but where $\mathbf{L}_{\hat{e}}$ is unknown, and not given by (34.20). Indeed, even small errors in computing the position of points in the image can lead to pose errors which will significantly impact the accuracy and the stability of the system (Fig. 34.11).

Avidbots Ex1016
Page 887 of 2154

34.4 Advanced Approaches

34.4.1 Hybrid VS

Suppose we have access to a control law for $\boldsymbol{\omega}_c$, such as the one used in PBVS ((34.23) or (34.25))

$$\boldsymbol{\omega}_c = -\lambda\, \theta \boldsymbol{u}\,. \tag{34.27}$$

How could we use this in conjunction with IBVS?

Considering a feature vector $\boldsymbol{s}_t$ and an error $\boldsymbol{e}_t$ devoted to control the translational degrees of freedom, we can partition the interaction matrix as follows

$$\begin{aligned}\dot{\boldsymbol{s}}_t &= \mathbf{L}_{\boldsymbol{s}_t} \boldsymbol{v}_c \\ &= \begin{pmatrix}\mathbf{L}_v & \mathbf{L}_{\boldsymbol{\omega}}\end{pmatrix}\begin{pmatrix}\boldsymbol{v}_c \\ \boldsymbol{\omega}_c\end{pmatrix} \\ &= \mathbf{L}_v \boldsymbol{v}_c + \mathbf{L}_{\boldsymbol{\omega}} \boldsymbol{\omega}_c\,.\end{aligned}$$

Now, setting $\dot{\boldsymbol{e}}_t = -\lambda \boldsymbol{e}_t$, we can solve for the desired translational control input as

$$\begin{aligned}-\lambda \boldsymbol{e}_t = \dot{\boldsymbol{e}}_t = \dot{\boldsymbol{s}}_t = \mathbf{L}_v \boldsymbol{v}_c + \mathbf{L}_{\boldsymbol{\omega}} \boldsymbol{\omega}_c\,, \\ \Rightarrow \boldsymbol{v}_c = -\mathbf{L}_v^{+}(\lambda \boldsymbol{e}_t + \mathbf{L}_{\boldsymbol{\omega}} \boldsymbol{\omega}_c)\,.\end{aligned} \tag{34.28}$$

We can think of the quantity $(\lambda \boldsymbol{e}_t + \mathbf{L}_{\boldsymbol{\omega}} \boldsymbol{\omega}_c)$ as a modified error term, one that combines the original error with the error that would be induced by the rotational motion due to $\boldsymbol{\omega}_c$. The translational control input $\boldsymbol{v}_c = -\mathbf{L}_v^{+}(\lambda \boldsymbol{e}_t + \mathbf{L}_{\boldsymbol{\omega}} \boldsymbol{\omega}_c)$ will drive this error to zero. The method known as 2.5-D visual servo [34.36] was the first to exploit such a partitioning in combining IBVS and PBVS. More precisely, in [34.36], $\boldsymbol{s}_t$ has been selected as the coordinates of an image point, and the logarithm of its depth, so that $\mathbf{L}_v$ is a triangular always invertible matrix. More precisely, we have $\boldsymbol{s}_t = (\boldsymbol{x}, \log Z)$, $\boldsymbol{s}_t^* = (\boldsymbol{x}^*, \log Z^*)$, $\boldsymbol{e}_t = (\boldsymbol{x} - \boldsymbol{x}^*, \log \rho_Z)$ where $\rho_Z = Z/Z^*$, and

$$\begin{aligned}\mathbf{L}_v &= \frac{1}{Z^* \rho_Z}\begin{pmatrix}-1 & 0 & x \\ 0 & -1 & y \\ 0 & 0 & -1\end{pmatrix}\mathbf{L}_{\boldsymbol{\omega}} \\ &= \begin{pmatrix}xy & -(1+x^2) & y \\ 1+y^2 & -xy & -x \\ -y & x & 0\end{pmatrix}\,.\end{aligned}$$

Note that the ratio ρ_Z can be obtained directly from the partial pose estimation algorithm that will be described in Sect. 34.6.

If we come back to the usual global representation of visual servo control schemes, we have $\boldsymbol{e} = (\boldsymbol{e}_t, \theta \boldsymbol{u})$ and $\mathbf{L}_e$ given by

$$\mathbf{L}_e = \begin{pmatrix}\mathbf{L}_v & \mathbf{L}_{\boldsymbol{\omega}} \\ \mathbf{0} & \mathbf{L}_{\theta u}\end{pmatrix}\,,$$

from which we immediately obtain the control law (34.27) and (34.28) by applying (34.5).

The behavior obtained using this choice for $\boldsymbol{s}_t$ is shown in Fig. 34.12 and VIDEO 64. Here, the point that has been considered in $\boldsymbol{s}_t$ is the center of gravity $\boldsymbol{x}_g$ of the target. We note the image trajectory of that point, which is a straight line as expected, and the nice decreasing of the camera velocity components, which makes this scheme very similar to the first PBVS one.

As for stability, it is clear that this scheme is globally asymptotically stable in perfect conditions. Furthermore, thanks to the triangular form of the interaction matrix $\mathbf{L}_e$, it is possible to analyze the stability of this scheme in the presence of calibration errors using the partial pose-estimation algorithm that will be described in Sect. 34.6 [34.37]. Finally, the only unknown constant parameter involved in this scheme, that is Z^*, can be estimated online using adaptive techniques [34.38].

Other hybrid schemes can be designed. For instance, in [34.39], the third component of $\boldsymbol{s}_t$ is different and has been selected so that all the target points remain in the camera field of view as far as possible. Another example has been proposed in [34.40]. In that case, $\boldsymbol{s}$ is selected as $\boldsymbol{s} = ({}^{c^*}\boldsymbol{t}_c, \boldsymbol{x}_g, \theta u_z)$ which provides with a block-triangular interaction matrix of the form

$$\mathbf{L}_e = \begin{pmatrix}\mathbf{R} & \mathbf{0} \\ \mathbf{L}'_v & \mathbf{L}'_{\boldsymbol{\omega}}\end{pmatrix}\,,$$

where $\mathbf{L}'_v$ and $\mathbf{L}'_{\boldsymbol{\omega}}$ can easily be computed. This scheme is such that, under perfect conditions, the camera trajectory is a straight line (since ${}^{c^*}\boldsymbol{t}_c$ is a part of $\boldsymbol{s}$), and the image trajectory of the center of gravity of the object is also a straight line (since $\boldsymbol{x}_g$ is also a part of $\boldsymbol{s}$). The translational camera degrees of freedom are devoted to realize the 3-D straight line, while the rotational camera degrees of freedom are devoted to realize the 2-D straight line and also compensate the 2-D motion of $\boldsymbol{x}_g$ due to the translational motion. As can be seen in Fig. 34.13 and VIDEO 65, this scheme is particularly satisfactory in practice.

Finally, it is possible to combine 2-D and 3-D features in different ways. For instance, in [34.41], it has been proposed to use in $\boldsymbol{s}$ the 2-D homogeneous coordinates of a set of image points expressed in pixels multiplied by their corresponding depth:

Avidbots Ex1016
Page 888 of 2154

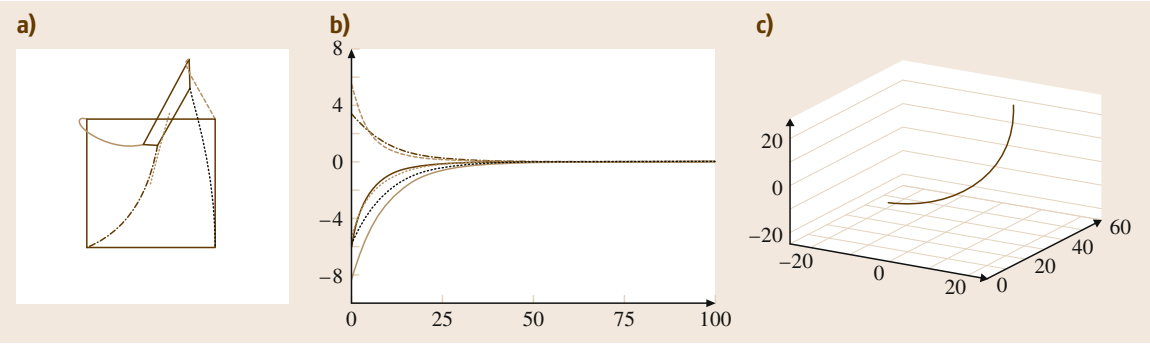


Fig. 34.12 2.5-D VS system behavior using $\boldsymbol{s} = (\mathbf{x}_g, \log(Z_g), \theta\boldsymbol{u})$ (refer to Fig. 34.2a-c for description)

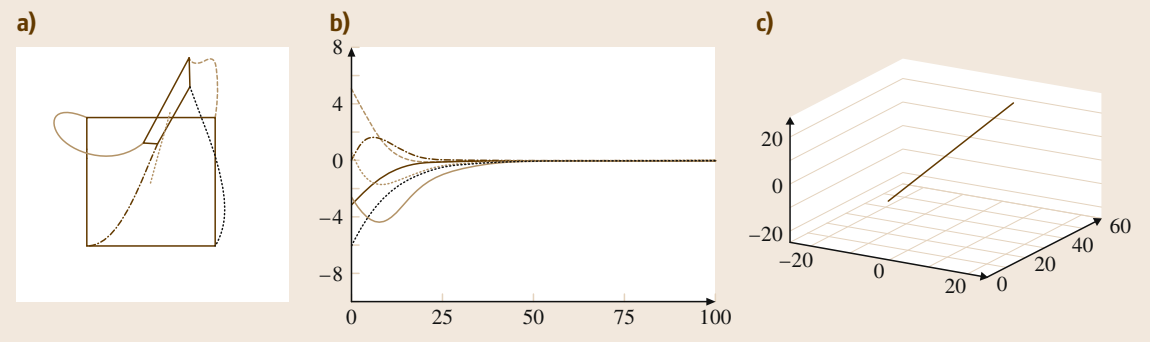


Fig. 34.13 2.5-D VS system behavior using $\boldsymbol{s} = ({}^{c^*}\mathbf{t}_c, \mathbf{x}_g, \theta u_z)$ (refer to Fig. 34.2a-c for description)

$\boldsymbol{s} = (u_1 Z_1, v_1 Z_1, Z_1, \cdots, u_n Z_n, v_n Z_n, Z_n)$. As for classical IBVS, we obtain in this case a set of redundant features, since at least three points have to be used to control the six camera degrees of freedom (here $k \geq 9$). However, it has been demonstrated in [34.42] that this selection of redundant features is free of attractive local minima.

34.4.2 Partitioned VS

The hybrid visual servo schemes described above have been designed to decouple the rotational motions from the translational ones by selecting adequate visual features defined in part in 2-D, and in part in 3-D (which is why they have been called 2.5-D visual servoing). This work has inspired some researchers to find features that exhibit similar decoupling properties but using only features expressed directly in the image. More precisely, the goal is to find six features such that each is related to only one degree of freedom (in which case the interaction matrix is a diagonal matrix). The Grail is to find a diagonal interaction matrix whose elements are constant, as near as possible to the identity matrix, leading to a pure, direct, and simple linear control problem.

The first work in this area partitioned the interaction matrix to isolate motion related to the optic axis [34.12]. Indeed, whatever the choice of $\boldsymbol{s}$, we have

$$\begin{aligned}\dot{\boldsymbol{s}} &= \mathbf{L}_{\boldsymbol{s}} \boldsymbol{v}_c \\ &= \mathbf{L}_{xy} \boldsymbol{v}_{xy} + \mathbf{L}_z \boldsymbol{v}_z \\ &= \dot{\boldsymbol{s}}_{xy} + \dot{\boldsymbol{s}}_z\end{aligned}$$

in which $\mathbf{L}_{xy}$ includes the first, second, fourth, and fifth columns of $\mathbf{L}_{\boldsymbol{s}}$, and $\mathbf{L}_z$ includes the third and sixth columns of $\mathbf{L}_{\boldsymbol{s}}$. Similarly, $\boldsymbol{v}_{xy} = (v_x, v_y, \omega_x, \omega_y)$ and $\boldsymbol{v}_z = (v_z, \omega_z)$. Here, $\dot{\boldsymbol{s}}_z = \mathbf{L}_z \boldsymbol{v}_z$ gives the component of $\dot{\boldsymbol{s}}$ due to the camera motion along and rotation about the optic axis, while $\dot{\boldsymbol{s}}_{xy} = \mathbf{L}_{xy} \boldsymbol{v}_{xy}$ gives the component of $\dot{\boldsymbol{s}}$ due to velocity along and rotation about the camera x and y axes.

Proceeding as above, by setting $\dot{\boldsymbol{e}} = -\lambda \boldsymbol{e}$ we obtain

$$-\lambda \boldsymbol{e} = \dot{\boldsymbol{e}} = \dot{\boldsymbol{s}} = \mathbf{L}_{xy} \boldsymbol{v}_{xy} + \mathbf{L}_z \boldsymbol{v}_z \,,$$

which leads to

$$\boldsymbol{v}_{xy} = -\mathbf{L}_{xy}^{+} [\lambda \boldsymbol{e}(t) + \mathbf{L}_z \boldsymbol{v}_z] \,.$$

As before, we can consider $[\lambda \boldsymbol{e}(t) + \mathbf{L}_z \boldsymbol{v}_z]$ as a modified error that incorporates the original error while taking into account the error that will be induced by $\boldsymbol{v}_z$.

Given this result, all that remains is to choose $\boldsymbol{s}$ and $\boldsymbol{v}_z$. As for basic IBVS, the coordinates of a collection of image points can be used in $\boldsymbol{s}$, while two new image features can be defined to determine $\boldsymbol{v}_z$:

- Define α, with $0 \leq \alpha < 2\pi$, as the angle between the horizontal axis of the image plane and the directed line segment joining two feature points.
  It is clear that α is closely related to the rotation around the optic axis.
- Define σ^2 to be the area of the polygon defined by these points. Similarly, σ^2 is closely related to the translation along the optic axis.

Using these features, $\boldsymbol{v}_z$ has been defined in [34.12] as

$$\begin{cases} v_z &= \lambda_{v_z} \ln \frac{\sigma^*}{\sigma} \,, \\ \omega_z &= \lambda_{\omega_z}(\alpha^* - \alpha) \,. \end{cases}$$

34.5 Performance Optimization and Planning

In some sense, partitioned methods represent an effort to optimize system performance by assigning distinct features and controllers to individual degrees of freedom. In this way, the designer performs a sort of offline optimization when allocating controllers to degrees of freedom. It is also possible to explicitly design controllers that optimize various system performance measures. We describe a few of these in this section.

34.5.1 Optimal Control and Redundancy Framework

An example of such an approach is given in [34.43] and [34.44], in which linear quadratic Gaussian (LQG) control design is used to choose gains that minimize a linear combination of state and control inputs. This approach explicitly balances the trade-off between tracking errors (since the controller attempts to drive $\boldsymbol{s} - \boldsymbol{s}^*$ to zero) and robot motion. A similar control approach is proposed in [34.45] where joint limit avoidance is considered simultaneously with the positioning task.

It is also possible to formulate optimality criteria that explicitly express the observability of robot motion in the image. For example, the singular value decomposition of the interaction matrix reveals which degrees of freedom are most apparent and can thus be easily controlled, while the condition number of the interaction matrix gives a kind of global measure of the visibility of motion. This concept has been called resolvability in [34.46] and motion perceptibility in [34.47]. By selecting features and designing controllers that maximize these measures, either along specific degrees of freedom or globally, the performance of the visual servo system can be improved.

The constraints considered to design the control scheme using the optimal control approach may be contradictory in some cases, leading the system to fail due to local minima in the objective function to be minimized. For example, it may happen that the motion produced to move away from a robot joint limit is exactly the opposite of the motion produced toward the desired pose, which results in a zero global motion. To avoid this potential problem, it is possible to use the gradient projection method, which is classical in robotics. Applying this method to visual servoing has been proposed in [34.3, 19]. The approach consists of projecting the secondary constraints $\boldsymbol{e}_s$ onto the null space of the vision-based task $\boldsymbol{e}$ so that they have no effect on the regulation of $\boldsymbol{e}$ to 0

$$\boldsymbol{e}_{\mathrm{g}} = \widehat{\mathbf{L}_e^+}\boldsymbol{e} + \mathbf{P}_e\,\boldsymbol{e}_s \,,$$

where $\boldsymbol{e}_{\mathrm{g}}$ is the new global task considered and

$$\mathbf{P}_e = \left(\mathbf{I}_6 - \widehat{\mathbf{L}_e^+}\widehat{\mathbf{L}_e}\right)$$

is such that

$$\widehat{\mathbf{L}_e}\mathbf{P}_e\boldsymbol{e}_s = 0 \,, \forall \boldsymbol{e}_s \,.$$

Avoiding the robot joint limits using this approach has been presented in [34.48]. However, when the vision-based task constrains all the camera degrees of freedom, the secondary constraints cannot be considered since, when $\widehat{\mathbf{L}_e}$ is of full rank 6, we have $\mathbf{P}_e\boldsymbol{e}_s = 0, \forall \boldsymbol{e}_s$. In this case, it is necessary to insert the constraints into a global objective function, such as navigation functions that are free of local minima [34.49, 50].

34.5.2 Switching Schemes

The partitioned methods described previously attempt to optimize performance by assigning individual controllers to specific degrees of freedom. Another way to use multiple controllers to optimize performance is to design switching schemes that select at each moment

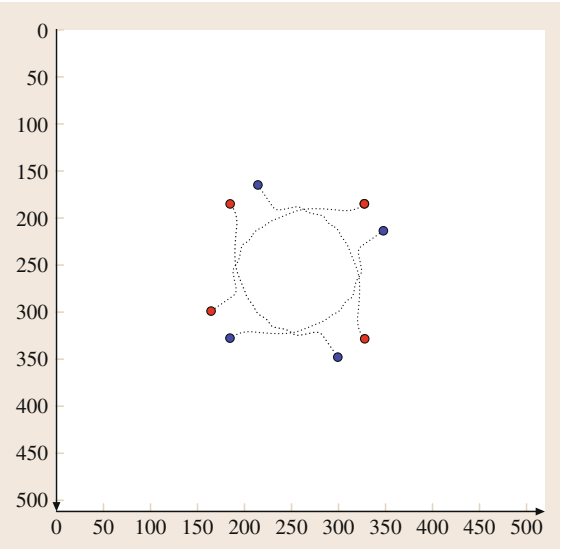


Fig. 34.14 Image feature trajectories for a rotation of 160° about the optical axis using a switched control scheme (initial point positions in *blue*, and desired points position in *red*)

in time which controller to use based on criteria to be optimized.

A simple switching controller can be designed using an IBVS and a PBVS controller as follows [34.51]. Let the system begin by using the IBVS controller. Consider the Lyapunov function for the PBVS controller given by $\mathcal{L}_P = \frac{1}{2}\|\boldsymbol{e}_P(t)\|^2$, with $\boldsymbol{e}_P(t) = ({}^c\boldsymbol{t}_o - {}^{c^*}\boldsymbol{t}_o, \theta\boldsymbol{u})$. If at any time the value of this Lyapunov function exceeds a threshold γ_P, the system switches to the PBVS controller. While using the PBVS controller, if at any time the value of the Lyapunov function $\mathcal{L}_I$ for the IBVS controller exceeds a threshold, $\mathcal{L}_I = \frac{1}{2}\|\boldsymbol{e}_I(t)\|^2 > \gamma_I$, the system switches to the IBVS controller. With this scheme, when the Lyapunov function for a particular controller exceeds a threshold, that controller is invoked, which in turn reduces the value of the corresponding Lyapunov function. If the switching thresholds are selected appropriately, the system is able to exploit the relative advantages of IBVS and PBVS, while avoiding their shortcomings.

An example of such a system is shown in Fig. 34.14 for the case of a rotation by 160° about the optical axis. Note that the system begins in IBVS mode and the features initially move on straight lines toward their goal positions in the image. However, as the camera retreats, the system switches to PBVS, which allows the camera to reach its desired position by combining a rotational motion around its optic axis and a forward translational motion, producing the circular trajectories observed in the image.

Other examples of temporal switching schemes can be found, such as the one developed in [34.52], to ensure the visibility of the target observed.

34.5.3 Feature Trajectory Planning

It is also possible to treat the optimization problem offline, during a planning stage, if we have sufficient knowledge of the system and world. In this case, several constraints can be taken into account simultaneously, such as obstacle avoidance [34.53], joint limit and occlusions avoidance, and ensuring the visibility of the target [34.54]. The feature trajectories $\boldsymbol{s}^*(t)$ that allow the camera to reach its desired pose while ensuring that the constraints are satisfied are determined using path planning techniques, such as the well-known potential field approach [34.54] or linear matrix inequality optimizations [34.55].

Coupling path planning with trajectory following also allows the robustness of the visual servo with respect to modeling errors to be significantly improved. Indeed, modeling errors may have large effects when the error $\boldsymbol{s}-\boldsymbol{s}^*$ is large, but have little effect when $\boldsymbol{s}-\boldsymbol{s}^*$ is small. Once the desired features trajectories $\boldsymbol{s}^*(t)$ such that $\boldsymbol{s}^*(0) = \boldsymbol{s}(0)$ have been designed during the planning stage, it is easy to adapt the control scheme to take into account the fact that $\boldsymbol{s}^*$ is varying, and to make the error $\boldsymbol{s}-\boldsymbol{s}^*$ remain small. More precisely, we now have

$$\dot{\boldsymbol{e}} = \dot{\boldsymbol{s}} - \dot{\boldsymbol{s}}^* = \mathbf{L}_{\boldsymbol{e}}\boldsymbol{v}_c - \dot{\boldsymbol{s}}^* ,$$

from which we deduce, by selecting as usual $\dot{\boldsymbol{e}} = -\lambda\boldsymbol{e}$ as the desired behavior,

$$\boldsymbol{v}_c = -\lambda\widehat{\mathbf{L}_{\boldsymbol{e}}^+}\boldsymbol{e} + \widehat{\mathbf{L}_{\boldsymbol{e}}^+}\dot{\boldsymbol{s}}^* .$$

The new second term of this control law anticipates the variation of $\boldsymbol{s}^*$, removing the tracking error it would produce. We will see in the 34.8 that the form of the control law is similar when tracking of a moving target is considered.

Avidbots Ex1016
Page 891 of 2154

34.6 Estimation of 3-D Parameters

All the control schemes described in the previous sections use 3-D parameters that are not directly available from the image measurements. As for IBVS, we recall that the range of the object with respect to the camera appears in the coefficients of the interaction matrix related to the translational degrees of freedom. Noticeable exceptions are the schemes based on a numerical estimation of $\mathbf{L}_e$ or of $\mathbf{L}_e^+$ (Sect. 34.2.8). Another exception is the IBVS scheme that uses the constant matrix $\widehat{\mathbf{L}_{e^*}^+}$ in the control scheme, in which only the depth for the desired pose is required, which is not so difficult to obtain in practice. As for PBVS and hybrid schemes that combine 2-D and 3-D data in $\boldsymbol{e}$, 3-D parameters appear both in the error $\boldsymbol{e}$ and in the interaction matrix. A correct estimation of the 3-D parameters involved is thus important for IBVS since they will have an effect on the camera motion during the task execution (they appear in the stability conditions (34.13) and (34.14)), while a correct estimation is crucial in PBVS and hybrid schemes since they will have also an effect on the accuracy of the pose reached after convergence.

If a calibrated stereo vision system is used, all 3-D parameters can be easily determined by triangulation, as mentioned in Sect. 34.2.5 and described in Chap. 32. Similarly, if a 3-D model of the object is known, all 3-D parameters can be computed from a pose-estimation algorithm. However, we recall that such an estimation can be quite unstable due to image noise (Sect. 34.3). As already said, IBVS does not require full pose estimation, simply the range of the object with respect to the camera. When image points are involved in $\boldsymbol{s}$, the range is expressed as the scalar depth Z or distance R of the corresponding world points, which appears in the interaction matrix (34.12), (34.16) and (34.17). This can be considered as a parameter estimation problem, that is, estimating the 3-D parameters from knowledge of the analytical form of the interaction matrix, and measurements of camera motion and visual feature position and velocity [34.56–58].

It is also possible to estimate 3-D parameters by using the epipolar geometry that relates the images of the same scene observed from different viewpoints. Indeed, in visual servoing, two images are generally available: the current one and the desired one. Given a set of matches between the image measurements in the current image and in the desired one, the fundamental matrix, or the essential matrix if the camera is calibrated, can be recovered [34.6], and then used in visual servoing [34.59]. Indeed, from the essential matrix, the rotation and the translation up to a scalar factor between the two views can be estimated. However, near the convergence of the visual servo, that is, when the current and desired images are similar, the epipolar geometry becomes degenerate and it is not possible to estimate accurately the partial pose between the two views. For this reason, using homography is generally preferred.

Let $\boldsymbol{x}_i$ and $\boldsymbol{x}_i^*$ denote the homogeneous image coordinates for a point in the current and desired images. Then $\boldsymbol{x}_i$ is related to $\boldsymbol{x}_i^*$ by

$$\boldsymbol{x}_i = \mathbf{H}_i \boldsymbol{x}_i^*$$

in which $\mathbf{H}_i$ is a homography matrix.

If all feature points lie on a 3-D plane, then there is a single homography matrix $\mathbf{H}$ such that $\boldsymbol{x}_i = \mathbf{H}\boldsymbol{x}_i^*$ for all i. This homography can be estimated using the position of four matched points in the desired and the current images. If all the features points do not belong to the same 3-D plane, then three points can be used to define such a plane and five supplementary points are needed to estimate $\mathbf{H}$ [34.60].

Once $\mathbf{H}$ is available, it can be decomposed as

$$\mathbf{H} = \mathbf{R} + \frac{\boldsymbol{t}}{d^*}\boldsymbol{n}^{*\mathrm{T}}, \tag{34.29}$$

in which $\mathbf{R}$ is the rotation matrix relating the orientation of the current and desired camera frames, $\boldsymbol{n}^*$ is the normal to the chosen 3-D plane expressed in the desired frame, d^* is the distance to the 3-D plane from the desired frame, and $\boldsymbol{t}$ is the translation between current and desired frames. From $\mathbf{H}$, it is thus possible to recover $\mathbf{R}$, $\boldsymbol{t}/d^*$, and $\boldsymbol{n}$. In fact, two solutions for these quantities exist [34.61], but it is quite easy to select the correct one using some knowledge about the desired pose. It is also possible to estimate the depth of any target point up to a common scale factor [34.54]. The unknown depth of each point that appears in classical IBVS can thus be expressed as a function of a single, constant parameter whatever the number of points. Similarly, the pose parameters required by PBVS can be recovered up to a scalar factor as for the translation term. The PBVS schemes described previously can thus be revisited using this approach, with the new error defined as the translation up to a scalar factor and the angle/axis parameterization of the rotation. This approach has also been used for the hybrid visual servoing schemes described in Sect. 34.4.1. In that case, using such homography estimation, it has been possible to analyze the stability of hybrid visual servoing schemes in the presence of calibration errors [34.36]. Finally, it is also possible to directly use the homography in the control scheme, avoiding thus its decomposition as a partial pose [34.62].

Avidbots Ex1016
Page 892 of 2154

34.7 Determining s^* and Matching Issues

All visual servo methods require knowledge of the desired feature values $\boldsymbol{s}^*$ which implicitly define constraints on the desired camera or robot pose with respect to the target. Three common approaches are employed. The first is when the task is directly specified as a desired value of some features to be reached. This is the case for instance when a target has to be centered in the image. This is also the case when the task is specified as a particular pose to reach and a PBVS is chosen. In that case however, the camera will reach its desired pose only if the camera is perfectly calibrated. Indeed, a coarse camera calibration will induce a biased estimation of the pose, which will make the final pose different from the desired one.

For IBVS and hybrid schemes, the second approach is to use knowledge of the object and camera projection models to compute $\boldsymbol{s}^*$ for the desired relative pose. Once again, the accuracy of the system directly depends on the camera calibration, since the camera intrinsic parameters are involved to compute $\boldsymbol{s}^*$.

Finally, the third approach is to simply record the feature values $\boldsymbol{s}^*$ when the camera or robot has the desired pose with respect to the target. This is usually done during an off-line teaching step. When it is possible in practice, this approach is very efficient since the positioning accuracy does not depend anymore on the camera calibration. This is still true for PBVS since, even if the pose estimated from the desired image is biased due to calibration errors, the same biased pose will be estimated once the robot will have converged so that the final image acquired by the camera will be the desired one.

So far we have not commented on the matching issues involved in visual servo methods. Two cases can be differentiated in function of the nature of the components of $\boldsymbol{s}$. When some components of $\boldsymbol{s}$ come from a pose estimation, it is necessary to match the measurements in the image (usually some image points) to the model of the object (usually some world points). Incorrect association will lead to an erroneous estimation of the pose. In all other cases, the calculation of the error vector $\boldsymbol{e}$ defined in (34.1) necessitates a matching between the measurements $\boldsymbol{m}(t)$ in the current image and $\boldsymbol{m}^*$ in the desired image. For instance, in all IBVS examples presented above, the camera observes four image points and we need to determine which of the four desired points to associate with each observed point so that the visual features $\boldsymbol{s}$ are correctly associated to their desired value $\boldsymbol{s}^*$. Incorrect association will lead to incorrect final camera pose and possibly a configuration which can not be achieved from any real camera pose. If we want to use the epipolar geometry or estimate a homography matrix, a similar matching process is necessary (Sect. 34.6).

This matching process is a classical computer vision problem. Note that it may be particularly difficult for the very first image, especially when the robot displacement to achieve is large, which generally implies large disparities between the initial and desired images. Once the association has been correctly performed for the very first image, the matching is greatly simplified since it transforms to a visual tracking problem where the results obtained for the previous image can be used as initialization for the current one.

34.8 Target Tracking

We now consider the case of a moving target and a constant desired value $\boldsymbol{s}^*$ for the features, the generalization to varying desired features $\boldsymbol{s}^*(t)$ being immediate. The time variation of the error is now given by

$$\dot{\boldsymbol{e}} = \mathbf{L}_{\boldsymbol{e}} \boldsymbol{v}_{\mathrm{c}} + \frac{\partial \boldsymbol{e}}{\partial t} , \tag{34.30}$$

where the term $\partial \boldsymbol{e}/\partial t$ expresses the time variation of $\boldsymbol{e}$ due to the generally unknown target motion. If the control law is still designed to try to ensure an exponential decoupled decrease of $\boldsymbol{e}$ (that is, once again $\dot{\boldsymbol{e}} = -\lambda \boldsymbol{e}$), we now obtain using (34.30)

$$\boldsymbol{v}_{\mathrm{c}} = -\lambda \widehat{\mathbf{L}_{\boldsymbol{e}}^{+}} \boldsymbol{e} - \widehat{\mathbf{L}_{\boldsymbol{e}}^{+}} \widehat{\frac{\partial \boldsymbol{e}}{\partial t}} , \tag{34.31}$$

where $\widehat{\partial \boldsymbol{e}/\partial t}$ is an estimation or an approximation of $\partial \boldsymbol{e}/\partial t$. This term must be introduced into the control law to compensate for the target motion.

Closing the loop, that is, inserting (34.31) into (34.30), we obtain

$$\dot{\boldsymbol{e}} = -\lambda \mathbf{L}_{\boldsymbol{e}} \widehat{\mathbf{L}_{\boldsymbol{e}}^{+}} \boldsymbol{e} - \mathbf{L}_{\boldsymbol{e}} \widehat{\mathbf{L}_{\boldsymbol{e}}^{+}} \widehat{\frac{\partial \boldsymbol{e}}{\partial t}} + \frac{\partial \boldsymbol{e}}{\partial t} . \tag{34.32}$$

Even if $\mathbf{L}_{\boldsymbol{e}} \widehat{\mathbf{L}_{\boldsymbol{e}}^{+}} > 0$, the error will converge to zero only if the estimation of $\widehat{\partial \boldsymbol{e}/\partial t}$ is sufficiently accurate so that

$$\mathbf{L}_{\boldsymbol{e}} \widehat{\mathbf{L}_{\boldsymbol{e}}^{+}} \widehat{\frac{\partial \boldsymbol{e}}{\partial t}} = \frac{\partial \boldsymbol{e}}{\partial t} , \tag{34.33}$$

Avidbots Ex1016
Page 893 of 2154

otherwise tracking errors will be observed. Indeed, by just solving the scalar differential equation $\dot{e} = -\lambda e + b$, which is a simplification of (34.32), we obtain $e(t) = e(0)\exp(-\lambda t) + b/\lambda$, which converges towards b/λ. On one hand, setting a high gain λ will reduce the tracking error, but on the other hand, setting the gain too high can make the system unstable. It is thus necessary to make b as small as possible.

Of course, if the system is known to be such that $\partial \boldsymbol{e}/\partial t = 0$ (that is, the camera observes a motionless object, as described in Sect. 34.1), no tracking error will appear with the most simple estimation given by $\widehat{\partial \boldsymbol{e}/\partial t} = 0$. Otherwise, a classical method in automatic control to cancel tracking errors consists of compensating the target motion through an integral term in the control law. In this case, wehave

$$\widehat{\frac{\partial \boldsymbol{e}}{\partial t}} = \mu \sum_j \boldsymbol{e}(j) \,,$$

where μ is the integral gain that has to be tuned. This scheme allows the tracking errors to be canceled only if the target has a constant velocity. Other methods, based on feedforward control, estimate the term $\widehat{\partial \boldsymbol{e}/\partial t}$ directly through the image measurements and the camera velocity, when it is available. Indeed, from (34.30), we obtain

$$\widehat{\frac{\partial \boldsymbol{e}}{\partial t}} = \widehat{\dot{\boldsymbol{e}}} - \widehat{\mathbf{L}_e}\,\widehat{\boldsymbol{v}_c} \,,$$

where $\widehat{\dot{\boldsymbol{e}}}$ can, for instance, be obtained as $\widehat{\dot{\boldsymbol{e}}}(t) = [\boldsymbol{e}(t) - \boldsymbol{e}(t-\Delta t)]/\Delta t$, Δt being the duration of the control loop. A Kalman filter [34.63] or more-elaborate filtering methods [34.64] can then be used to improve the estimated values obtained. If some knowledge about the target velocity or the target trajectory is available, it can of course be used to smooth or predict the motion [34.65–67]. For instance, in [34.68], the periodic motion of the heart and breathing are compensated for an application of visual servoing in medical robotics. Finally, other methods have been developed to remove the perturbations induced by the target motion as fast as possible [34.43], using for instance predictive controllers [34.69].

34.9 Eye-in-Hand and Eye-to-Hand Systems Controlled in the Joint Space

In the previous sections, we have considered the six components of the camera velocity as the input of the robot controller. As soon as the robot is not able to realize this motion, for instance, because it has fewer than six degrees of freedom, the control scheme must be expressed in the joint space. In this section, we describe how this can be done, and in the process develop a formulation for eye-to-hand systems.

In the joint space, the system equations for both the eye-to-hand and eye-in-hand configurations have the same form

$$\dot{\boldsymbol{s}} = \mathbf{J}_s\,\dot{\boldsymbol{q}} + \frac{\partial \boldsymbol{s}}{\partial t} \,. \tag{34.34}$$

Here, $\mathbf{J}_s \in \mathbb{R}^{k\times n}$ is the feature Jacobian matrix, which can be linked to the interaction matrix, and n is the number of robot joints.

For an eye-in-hand system (Fig. 34.15a), $\partial \boldsymbol{s}/\partial t$ is the time variation of $\boldsymbol{s}$ due to a potential object motion, and $\mathbf{J}_s$ is given by

$$\mathbf{J}_s = \mathbf{L}_s\,{}^{c}\boldsymbol{X}_N \mathbf{J}(\boldsymbol{q}) \,, \tag{34.35}$$

where:

- ${}^{c}\boldsymbol{X}_N$ is the spatial motion transform matrix (as defined in Chap. 02 and recalled in (34.15)) from the vision sensor frame to the end-effector frame. It is usually a constant matrix (as long as the vision sensor is rigidly attached to the end-effector). Thanks to the robustness of closed-loop control schemes, a coarse approximation of this transform matrix is sufficient in visual servoing. If needed, an accurate estimation is possible through classical hand–eye calibration methods [34.70].
- $\mathbf{J}(\boldsymbol{q})$ is the robot Jacobian expressed in the end-effector frame (as defined in Chap. 02)

For an eye-to-hand system (Fig. 34.15b), $\partial \boldsymbol{s}/\partial t$ is now the time variation of $\boldsymbol{s}$ due to a potential vision sensor motion and $\mathbf{J}_s$ can be expressed as

$$\mathbf{J}_s = -\mathbf{L}_s\,{}^{c}\boldsymbol{X}_N\,{}^{N}\mathbf{J}(\boldsymbol{q}) \,, \tag{34.36}$$

$$= -\mathbf{L}_s\,{}^{c}\boldsymbol{X}_0\,{}^{0}\mathbf{J}(\boldsymbol{q}) \,. \tag{34.37}$$

In (34.36), the classical robot Jacobian ${}^{N}\mathbf{J}(\boldsymbol{q})$ expressed in the end-effector frame is used but the spatial motion transform matrix ${}^{c}\boldsymbol{X}_N$ from the vision sensor frame to the end-effector frame changes all along the robot mo-

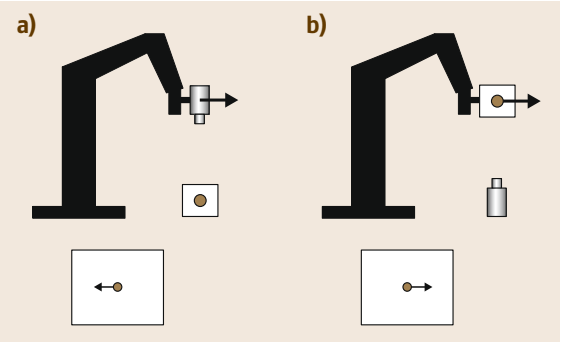


Fig.34.15a,b **(a)** Eye-in-hand system, **(b)** eye-to-hand system: system schematic (*top*) and opposite image motion produced by the same robot motion (*bottom*)

tion, and it has to be estimated at each iteration of the control scheme, usually using pose-estimation methods.

In (34.37), the robot Jacobian ${}^0\mathbf{J}(\boldsymbol{q})$ is expressed in the robot reference frame, and the spatial motion transform matrix ${}^c\boldsymbol{X}_0$ from the vision sensor frame to that reference frame is constant as long as the camera does not move. In this case, which is convenient in practice, a coarse approximation of ${}^c\boldsymbol{X}_0$ is usually sufficient.

Once the modeling step is finished, it is quite easy to follow the procedure that has been used above to design a control scheme expressed in the joint space, and to determine the sufficient condition to ensure the stability of the control scheme. We obtain, considering again $\boldsymbol{e} = \boldsymbol{s} - \boldsymbol{s}^*$, and an exponential decoupled decrease of $\boldsymbol{e}$

$$\dot{\boldsymbol{q}} = -\lambda \widehat{\mathbf{J}_e^+} \boldsymbol{e} - \widehat{\mathbf{J}_e^+} \widehat{\frac{\partial \boldsymbol{e}}{\partial t}} \,. \tag{34.38}$$

If $k = n$, considering as in Sect. 34.1 the Lyapunov function $\mathcal{L} = \frac{1}{2}\|\boldsymbol{e}(t)\|^2$, a sufficient condition to ensure the global asymptotic stability is given by

$$\mathbf{J}_e \widehat{\mathbf{J}_e^+} > 0 \,. \tag{34.39}$$

If $k > n$, we obtain similarly to Sect. 34.1

$$\widehat{\mathbf{J}_e^+} \mathbf{J}_e > 0 \tag{34.40}$$

to ensure the local asymptotic stability of the system. Note that the actual extrinsic camera parameters appear in $\mathbf{J}_e$ while the estimated ones are used in $\widehat{\mathbf{J}_e^+}$. It is thus possible to analyze the robustness of the control scheme with respect to the camera extrinsic parameters. It is also possible to estimate directly the numerical value of $\mathbf{J}_e$ or $\mathbf{J}_e^+$ using the methods described in Sect. 34.2.8.

Finally, to remove tracking errors, we have to ensure that

$$\mathbf{J}_e \widehat{\mathbf{J}_e^+} \widehat{\frac{\partial \boldsymbol{e}}{\partial t}} = \frac{\partial \boldsymbol{e}}{\partial t} \,.$$

Let us note that, even if the robot has six degrees of freedom, it is generally not equivalent to first compute $\boldsymbol{v}_c$ using (34.5) and then deduce $\dot{\boldsymbol{q}}$ using the robot inverse Jacobian, and to compute directly $\dot{\boldsymbol{q}}$ using (34.38). Indeed, it may occur that the robot Jacobian $\mathbf{J}(\boldsymbol{q})$ is singular while the feature Jacobian $\mathbf{J}_s$ is not (that may occur when $k < n$). Furthermore, the properties of the pseudo-inverse ensure that using (34.5), $\|\boldsymbol{v}_c\|$ is minimal while using (34.38), $\|\dot{\boldsymbol{q}}\|$ is minimal. As soon as

$$\mathbf{J}_e^+ \neq \mathbf{J}^+(\boldsymbol{q})^N \boldsymbol{X}_c \mathbf{L}_e^+ \,,$$

the control schemes will be different and will induce different robot trajectories. The choice of the state space is thus important.

34.10 Under Actuated Robots

Many useful robots are under actuated, that is, they cannot move instantaneously in all directions because of the number or configuration of their actuators (Chap. 52). For instance, a quadrotor flying robot is under actuated since it has only four actuators while the dimension of its configuration space is six. Many other useful robots are subject to non-holonomic constraints (Chap. 49), leading to similar motion inability. A car for instance has only two degrees of freedom (for velocity and steering) while the dimension of its configuration space is three (position and orientation in the ground plane).

A consequence of under actuation is that time varying manoeuvres may be required in order to achieve particular goal states. For example, if a quadrotor has to move forward, it must first change its attitude, pitching down, so that a component of its thrust vector is able to accelerate the vehicle forward. For a visual servo system this can be problematic since the initial attitude change will affect the value of the visual features even before the vehicle has moved. In fact the attitude change increases the error and this is ultimately destabilising.

A common and expedient solution [34.71] is to *de-rotate* the image, that is, to use information from

Avidbots Ex1016
Page 895 of 2154

a non-vision attitude sensor such as an inertial measurement unit (IMU) (Chap. 29) to correct the feature coordinates as if they had been viewed by a virtual camera whose optical axis has a constant orientation in space, typically, straight down. As discussed in Sect. 34.2.8, the spherical projection model is well suited for such image transformation, thanks to its invariance properties with respect to rotational motion.

For a non-holonomic vehicle, there are several solutions. In the non-general case where the vehicle is able to follow a smooth path from its initial to goal configuration, a controller based on visually estimated relative pose (range, heading angle and lateral offset) can be used (Chap. 49), following a PBVS strategy. Particular controllers based on the epipolar geometry or the trifocal tensor have also been designed [34.72, 73]. For the more general case, switching control laws can be used. If possible in practice, another common solution is to add a controlled DOF between the vehicle and the camera in order to bypass the non-holonomic constraint through redundancy. It is thus possible to control the full camera configuration, but not the robot one.

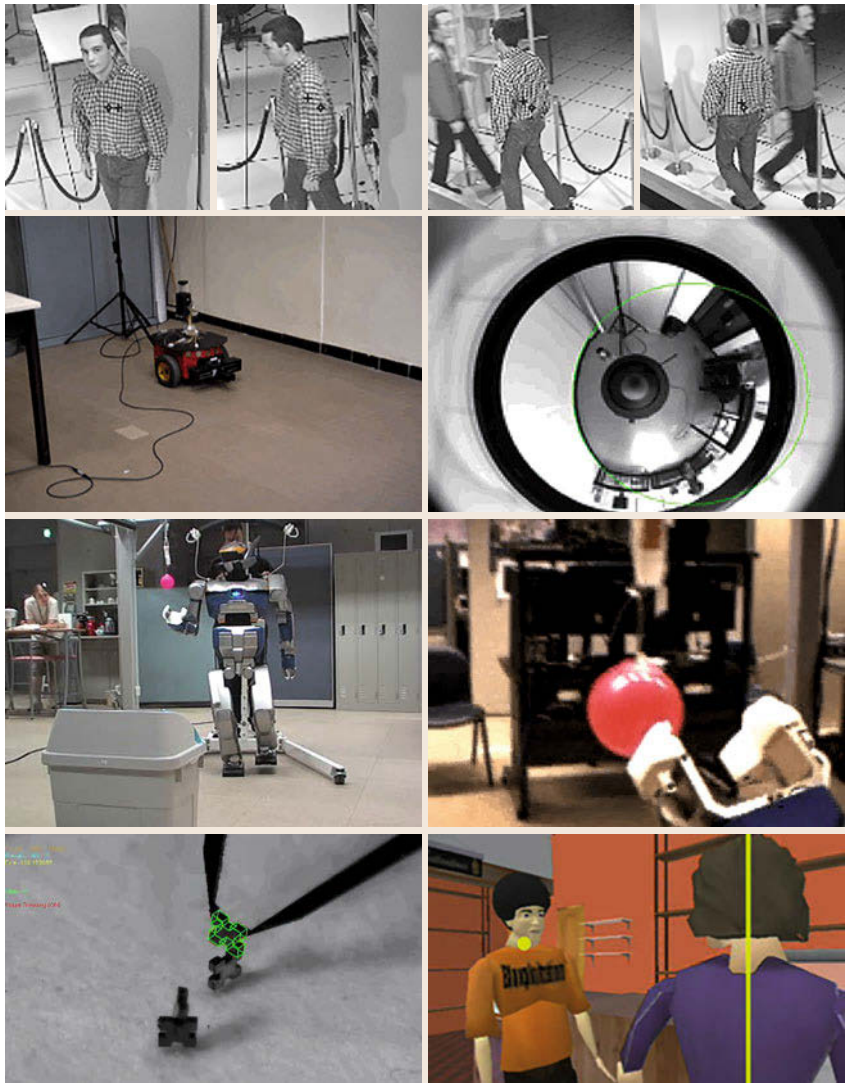


Fig. 34.16 Few applications of visual servoing: Gaze control for target tracking, navigation of a mobile robot to follow a wall using an omnidirectional vision sensor, grasping a ball with a humanoid robot, assembly of MEMS and film of a dialogue within the constraints of a script in animation

Avidbots Ex1016
Page 896 of 2154

34.11 Applications

Applications of visual servoing in robotics are numerous. It can be used as soon as a vision sensor is available and a task is assigned to a dynamic system to control its motion. A non exhaustive list of examples are (Fig. 34.16):

- The control of a pan-tilt-zoom camera for target tracking
- Grasping using a robot arm
- Locomotion and dextrous manipulation with a humanoid robot
- Micro or nano manipulation of microelectromechanical system (MEMS) or biological cells
- Accurate positioning of a parallel robot
- Pipe inspection by an underwater autonomous vehicle
- Autonomous navigation of a mobile robot in indoor or outdoor environment
- Aircraft landing
- Autonomous satellite rendezvous
- Biopsy using ultrasound probes or heart motion compensation in medical robotics
- Virtual cinematography in animation.

Software are available to help any user interested in developing visual servoing applications. We recommend the Robotics, Machine Vision Toolbox for Matlab [34.74] and the ViSP C++ library [34.11].

34.12 Conclusions

In this chapter we have only considered velocity controllers, which are convenient for most classical robot arms. However, the dynamics of the robot must of course be taken into account for high-speed tasks. Features related to the image motion [34.75], image intensity [34.76], or coming from other vision sensors (RGB-D sensors, ultrasonic probes [34.77], etc.) necessitate reconsideration of the modeling issues to select adequate visual features. Finally, fusing visual features with data coming from other sensors (force sensor, proximity sensors, etc.) at the level of the control scheme will allow new research topics to be addressed. The end of fruitful research in the field of visual servo is thus nowhere yet in sight.

Video-References

VIDEO 59 IBVS on a 6 DOF robot arm (1);
available from http://handbookofrobotics.org/view-chapter/34/videodetails/59

VIDEO 60 IBVS on a 6 DOF robot arm (2);
available from http://handbookofrobotics.org/view-chapter/34/videodetails/60

VIDEO 61 IBVS on a 6 DOF robot arm (3);
available from http://handbookofrobotics.org/view-chapter/34/videodetails/61

VIDEO 62 PBVS on a 6 DOF robot arm (1);
available from http://handbookofrobotics.org/view-chapter/34/videodetails/62

VIDEO 63 PBVS on a 6 DOF robot arm (2);
available from http://handbookofrobotics.org/view-chapter/34/videodetails/63

VIDEO 64 2.5-D VS on a 6 DOF robot arm (1);
available from http://handbookofrobotics.org/view-chapter/34/videodetails/64

VIDEO 65 2.5-D VS on a 6 DOF robot arm (2);
available from http://handbookofrobotics.org/view-chapter/34/videodetails/65

References

34.1 L. Weiss, A. Sanderson, C. Neuman: Dynamic sensor-based control of robots with visual feedback, IEEE J. Robot. Autom. **3**, 404–417 (1987)

34.2 S. Hutchinson, G. Hager, P. Corke: A tutorial on visual servo control, IEEE Trans. Robot. Autom. **12**, 651–670 (1996)

34.3 B. Espiau, F. Chaumette, P. Rives: A new approach to visual servoing in robotics, IEEE Trans. Robot. Autom. **8**, 313–326 (1992)

34.4 J. Feddema, O. Mitchell: Vision-guided servoing with feature-based trajectory generation, IEEE Trans. Robot. Autom. **5**, 691–700 (1989)

Avidbots Ex1016
Page 897 of 2154

34.5 D. Forsyth, J. Ponce: *Computer Vision: A Modern Approach* (Prentice Hall, Upper Saddle River 2003)
34.6 Y. Ma, S. Soatto, J. Kosecka, S. Sastry: *An Invitation to 3-D Vision: From Images to Geometric Models* (Springer, New York 2003)
34.7 P. Corke: *Robotics, Vision and Control: Fundamental Algorithms in MATLAB* (Springer, Berlin, Heidelberg 2011)
34.8 H. Michel, P. Rives: Singularities in the Determination of the Situation of a Robot Effector from the Perspective View of Three Points. Res. Rep. RR-1850 (INRIA 1993)
34.9 M. Fischler, R. Bolles: Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography, Communications ACM **24**, 381–395 (1981)
34.10 E. Malis: Improving vision-based control using efficient second-order minimization techniques, IEEE Int. Conf. Robot. Autom., New Orleans (2004) pp. 1843–1848
34.11 E. Marchand, F. Spindler, F. Chaumette: ViSP for visual servoing: A generic software platform with a wide class of robot control skills, IEEE Robot. Autom. Mag. **12**(4), 40–52 (2005), https://team.inria.fr/lagadic/visp/visp.html
34.12 P. Corke, S. Hutchinson: A new partitioned approach to image-based visual servo control, IEEE Trans. Robot. Autom. **17**, 507–515 (2001)
34.13 F. Chaumette: Potential problems of stability and convergence in image-based and position-based visual servoing, Lect. Note. Contr. Inform. Sci. **237**, 66–78 (1998)
34.14 E. Malis: Visual servoing invariant to changes in camera intrinsic parameters, IEEE Trans. Robot. Autom. **20**, 72–81 (2004)
34.15 A. Isidori: *Nonlinear Control Systems*, 3rd edn. (Springer, Berlin, Heidelberg 1995)
34.16 G. Hager, W. Chang, A. Morse: Robot feedback control based on stereo vision: Towards calibration-free hand-eye coordination, IEEE Control Syst. Mag. **15**, 30–39 (1995)
34.17 M. Iwatsuki, N. Okiyama: A new formulation of visual servoing based on cylindrical coordinate system, IEEE Trans. Robot. Autom. **21**, 266–273 (2005)
34.18 F. Chaumette, P. Rives, B. Espiau: Classification and realization of the different vision-based tasks, Robot. Autom. Syst. **7**, 199–228 (1993)
34.19 A. Castano, S. Hutchinson: Visual compliance: Task directed visual servo control, IEEE Trans. Robot. Autom. **10**, 334–342 (1994)
34.20 G. Hager: A modular system for robust positioning using feedback from stereo vision, IEEE Trans. Robot. Autom. **13**, 582–595 (1997)
34.21 F. Chaumette: Image moments: A general and useful set of features for visual servoing, IEEE Trans. Robot. Autom. **20**, 713–723 (2004)
34.22 O. Tahri, F. Chaumette: Point-based and region-based image moments for visual servoing of planar objects, IEEE Trans. Robot. **21**, 1116–1127 (2005)
34.23 C. Geyer, K. Daniilidis: Catadioptric projective geometry, Int. J. Comput. Vis. **45**(3), 223–243 (2001)
34.24 T. Hamel, R. Mahony: Visual servoing of an under-actuated dynamic rigid-body system: An image-based approach, IEEE Trans Robot. **18**(2), 187–198 (2002)
34.25 I. Suh: Visual servoing of robot manipulators by fuzzy membership function based neural networks. In: *Visual Servoing*, Robotics and Automated Systems, Vol. 7, ed. by K. Hashimoto (World Scientific, Singapore 1993) pp. 285–315
34.26 G. Wells, C. Venaille, C. Torras: Vision-based robot positioning using neural networks, Image Vis. Comput. **14**, 75–732 (1996)
34.27 J.T. Lapresté, F. Jurie, F. Chaumette: An efficient method to compute the inverse jacobian matrix in visual servoing, IEEE Int. Conf. Robot. Autom., New Orleans (2004) pp. 727–732
34.28 K. Hosada, M. Asada: Versatile visual servoing without knowledge of true jacobian, IEEE/RSJ Int. Conf. Intell. Robots Syst., München (1994) pp. 186–193
34.29 M. Jägersand, O. Fuentes, R. Nelson: Experimental evaluation of uncalibrated visual servoing for precision manipulation, IEEE Int. Conf. Robot. Autom., Albuquerque (1997) pp. 2874–2880
34.30 J. Piepmeier, G.M. Murray, H. Lipkin: Uncalibrated dynamic visual servoing, IEEE Trans. Robot. Autom. **20**, 143–147 (2004)
34.31 K. Deguchi: Direct interpretation of dynamic images and camera motion for visual servoing without image feature correspondence, J. Robot. Mechatron. **9**(2), 104–110 (1997)
34.32 W. Wilson, C. Hulls, G. Bell: Relative end-effector control using cartesian position based visual servoing, IEEE Trans. Robot. Autom. **12**, 684–696 (1996)
34.33 B. Thuilot, P. Martinet, L. Cordesses, J. Gallice: Position based visual servoing: Keeping the object in the field of vision, IEEE Int. Conf. Robot. Autom., Washington (2002) pp. 1624–1629
34.34 D. Dementhon, L. Davis: Model-based object pose in 25 lines of code, Int. J. Comput. Vis. **15**, 123–141 (1995)
34.35 D. Lowe: Three-dimensional object recognition from single two-dimensional images, Artif. Intell. **31**(3), 355–395 (1987)
34.36 E. Malis, F. Chaumette, S. Boudet: 2-1/2 D visual servoing, IEEE Trans. Robot. Autom. **15**, 238–250 (1999)
34.37 E. Malis, F. Chaumette: Theoretical improvements in the stability analysis of a new class of model-free visual servoing methods, IEEE Trans. Robot. Autom. **18**, 176–186 (2002)
34.38 J. Chen, D. Dawson, W. Dixon, A. Behal: Adaptive homography-based visual servo tracking for fixed camera-in-hand configurations, IEEE Trans. Control Syst. Technol. **13**, 814–825 (2005)
34.39 G. Morel, T. Leibezeit, J. Szewczyk, S. Boudet, J. Pot: Explicit incorporation of 2-D constraints in vision-based control of robot manipulators, Lect. Note. Contr. Inform. Sci. **250**, 99–108 (2000)

Avidbots Ex1016
Page 898 of 2154

34.40 F. Chaumette, E. Malis: 2 1/2 D visual servoing: a possible solution to improve image-based and position-based visual servoings, IEEE Int. Conf. Robot. Autom., San Fransisco (2000) pp. 630–635

34.41 E. Cervera, A.D. Pobil, F. Berry, P. Martinet: Improving image-based visual servoing with three-dimensional features, Int. J. Robot. Res. **22**, 821–840 (2004)

34.42 F. Schramm, G. Morel, A. Micaelli, A. Lottin: Extended 2-D visual servoing, IEEE Int. Conf. Robot. Autom., New Orleans (2004) pp. 267–273

34.43 N. Papanikolopoulos, P. Khosla, T. Kanade: Visual tracking of a moving target by a camera mounted on a robot: A combination of vision and control, IEEE Trans. Robot. Autom. **9**, 14–35 (1993)

34.44 K. Hashimoto, H. Kimura: LQ optimal and nonlinear approaches to visual servoing, Robot. Autom. Syst. **7**, 165–198 (1993)

34.45 B. Nelson, P. Khosla: Strategies for increasing the tracking region of an eye-in-hand system by singularity and joint limit avoidance, Int. J. Robot. Res. **14**, 225–269 (1995)

34.46 B. Nelson, P. Khosla: Force and vision resolvability for assimilating disparate sensory feedback, IEEE Trans. Robot. Autom. **12**, 714–731 (1996)

34.47 R. Sharma, S. Hutchinson: Motion perceptibility and its application to active vision-based servo control, IEEE Trans. Robot. Autom. **13**, 607–617 (1997)

34.48 E. Marchand, F. Chaumette, A. Rizzo: Using the task function approach to avoid robot joint limits and kinematic singularities in visual servoing, IEEE/RSJ Int. Conf. Intell. Robots Syst., Osaka (1996) pp. 1083–1090

34.49 E. Marchand, G. Hager: Dynamic sensor planning in visual servoing, IEEE Int. Conf. Robot. Autom., Leuven (1998) pp. 1988–1993

34.50 N. Cowan, J. Weingarten, D. Koditschek: Visual servoing via navigation functions, IEEE Trans. Robot. Autom. **18**, 521–533 (2002)

34.51 N. Gans, S. Hutchinson: An asymptotically stable switched system visual controller for eye in hand robots, IEEE/RSJ Int. Conf. Intell. Robots Syst., Las Vegas (2003) pp. 735–742

34.52 G. Chesi, K. Hashimoto, D. Prattichizio, A. Vicino: Keeping features in the field of view in eye-in-hand visual servoing: a switching approach, IEEE Trans. Robot. Autom. **20**, 908–913 (2004)

34.53 K. Hosoda, K. Sakamato, M. Asada: Trajectory generation for obstacle avoidance of uncalibrated stereo visual servoing without 3-D reconstruction, IEEE/RSJ Int. Conf. Intell. Robots Syst. 3, Pittsburgh (1995) pp. 29–34

34.54 Y. Mezouar, F. Chaumette: Path planning for robust image-based control, IEEE Trans. Robot. Autom. **18**, 534–549 (2002)

34.55 G. Chesi: Visual servoing path-planning via homogeneous forms and LMI optimizations, IEEE Trans. Robot. **25**(2), 281–291 (2009)

34.56 L. Matthies, T. Kanade, R. Szeliski: Kalman filter-based algorithms for estimating depth from image sequences, Int. J. Comput. Vis. **3**(3), 209–238 (1989)

34.57 C.E. Smith, N. Papanikolopoulos: Computation of shape through controlled active exploration, IEEE Int. Conf. Robot. Autom., San Diego (1994) pp. 2516–2521

34.58 A. De Luca, G. Oriolo, P. Robuffo Giordano: Feature depth observation for image-based visual servoing: Theory and experiments, Int. J. Robot. Res. **27**(10), 1093–1116 (2008)

34.59 R. Basri, E. Rivlin, I. Shimshoni: Visual homing: Surfing on the epipoles, Int. J. Comput. Vis. **33**, 117–137 (1999)

34.60 E. Malis, F. Chaumette, S. Boudet: 2 1/2 D visual servoing with respect to unknown objects through a new estimation scheme of camera displacement, Int. J. Comput. Vis. **37**, 79–97 (2000)

34.61 O. Faugeras: *Three-Dimensional Computer Vision: A Geometric Viewpoint* (MIT Press, Cambridge 1993)

34.62 G. Silveira, E. Malis: Direct visual servoing: Vision-based estimation and control using only nonmetric information, IEEE Trans. Robot. **28**(4), 974–980 (2012)

34.63 P. Corke, M. Goods: Controller design for high performance visual servoing, 12th World Congr. IFAC'93, Sydney (1993) pp. 395–398

34.64 F. Bensalah, F. Chaumette: Compensation of abrupt motion changes in target tracking by visual servoing, IEEE/RSJ Int. Conf. Intell. Robots Syst., Pittsburgh (1995) pp. 181–187

34.65 P. Allen, B. Yoshimi, A. Timcenko, P. Michelman: Automated tracking and grasping of a moving object with a robotic hand-eye system, IEEE Trans. Robot. Autom. **9**, 152–165 (1993)

34.66 K. Hashimoto, H. Kimura: Visual servoing with non linear observer, IEEE Int. Conf. Robot. Autom., Nagoya (1995) pp. 484–489

34.67 A. Rizzi, D. Koditschek: An active visual estimator for dexterous manipulation, IEEE Trans. Robot. Autom. **12**, 697–713 (1996)

34.68 R. Ginhoux, J. Gangloff, M. de Mathelin, L. Soler, M.A. Sanchez, J. Marescaux: Active filtering of physiological motion in robotized surgery using predictive control, IEEE Trans. Robot. **21**, 67–79 (2005)

34.69 J. Gangloff, M. de Mathelin: Visual servoing of a 6-DOF manipulator for unknown 3-D profile following, IEEE Trans. Robot. Autom. **18**, 511–520 (2002)

34.70 R. Tsai, R. Lenz: A new technique for fully autonomous efficient 3-D robotics hand-eye calibration, IEEE Trans. Robot. Autom. **5**, 345–358 (1989)

34.71 N. Guenard, T. Hamel, R. Mahony: A practical visual servo control for an unmanned aerial vehicle, IEEE Trans. Robot. **24**(2), 331–340 (2008)

34.72 G.L. Mariottini, G. Oriolo, D. Prattichizo: Image-based visual servoing for nonholonomic mobile robots using epipolar geometry, IEEE Trans. Robot. **23**(1), 87–100 (2007)

34.73 G. Lopez-Nicolas, J.J. Guerrero, C. Sagues: Visual control through the trifocal tensor for nonholonomic robots, Robot. Auton. Syst. **58**(2), 216–226 (2010)

Avidbots Ex1016
Page 899 of 2154

34.74 P. Corke: *Robotics, Vision and Control: Fundamental Algorithms in MATLAB*, Springer Tracts in Advanced Robotics, Vol. 73 (Springer, Berlin, Heidelberg 2011)
34.75 A. Crétual, F. Chaumette: Visual servoing based on image motion, Int. J. Robot. Res. **20**(11), 857–877 (2001)
34.76 C. Collewet, E. Marchand: Photometric visual servoing, IEEE Trans. Robot. **27**(4), 828–834 (2011)
34.77 R. Mebarki, A. Krupa, F. Chaumette: 2D ultrasound probe complete guidance by visual servoing using image moments, IEEE Trans. Robot. **26**(2), 296–306 (2010)

Avidbots Ex1016
Page 900 of 2154

Multimedia Contents

35. Multisensor Data Fusion

Hugh Durrant-Whyte, Thomas C. Henderson

Multisensor data fusion is the process of combining observations from a number of different sensors to provide a robust and complete description of an environment or process of interest. Data fusion finds wide application in many areas of robotics such as object recognition, environment mapping, and localization.

This chapter has three parts: methods, architectures, and applications. Most current data fusion methods employ probabilistic descriptions of observations and processes and use Bayes' rule to combine this information. This chapter surveys the main probabilistic modeling and fusion techniques including grid-based models, Kalman filtering, and sequential Monte Carlo techniques. This chapter also briefly reviews a number of nonprobabilistic data fusion methods. Data fusion systems are often complex combinations of sensor devices, processing, and fusion algorithms. This chapter provides an overview of key principles in data fusion architectures from both a hardware and algorithmic viewpoint. The applications of data fusion are pervasive in robotics and underly the core problem of sensing, estimation, and perception. We highlight two example applications that bring out these features. The first describes a navigation or self-tracking application for an autonomous vehicle. The second describes an application in mapping and environment modeling.

35.1 **Multisensor Data Fusion Methods** 867
35.1.1 Bayes' Rule 868
35.1.2 Probabilistic Grids 869
35.1.3 The Kalman Filter 872
35.1.4 Sequential Monte Carlo Methods 876
35.1.5 Alternatives to Probability 878
35.2 **Multisensor Fusion Architectures** 880
35.2.1 Architectural Taxonomy 880
35.2.2 Centralized, Local Interaction, and Hierarchical 882
35.2.3 Decentralized, Global Interaction, and Heterarchical 882
35.2.4 Decentralized, Local Interaction, and Hierarchical 883
35.2.5 Decentralized, Local Interaction, and Heterarchical 884
35.3 **Applications** 885
35.3.1 Dynamic System Control 885
35.3.2 ANSER II: Decentralized Data Fusion 888
35.3.3 Recent Developments 889
35.4 **Conclusions** 889
Video-References 889
References 890

The essential algorithmic tools of data fusion are reasonably well established. However, the development and use of these tools in realistic robotics applications is still developing.

35.1 Multisensor Data Fusion Methods

The most widely used data fusion methods employed in robotics originate in the fields of statistics, estimation, and control. However, the application of these methods in robotics has a number of unique features and challenges. In particular, autonomy is most often the goal and so results must be presented and interpreted in a form from which autonomous decisions can be made, for recognition or navigation, for example.

In this section we review the main data fusion methods employed in robotics. These are very often based on probabilistic methods, which are indeed now considered the standard approach to data fusion in all robotics applications [35.1]. Probabilistic data fusion methods are generally based on Bayes' rule for combining prior and observation information. Practically, this may be implemented in a number of ways: through the use of the Kalman and extended Kalman filters, through sequential Monte Carlo methods, or through the use of functional density estimates. Each of these is reviewed. There are a number of alternatives to probabilistic methods. These include the theory of evidence and interval methods. Such alternative techniques are not as widely used as they once were, however they have some special features that can be advantageous in specific problems. These, too, are briefly reviewed.

35.1.1 Bayes' Rule

Bayes' rule lies at the heart of most data fusion methods. In general, Bayes' rule provides a means to make inferences about an object or environment of interest described by a state $\boldsymbol{x}$, given an observation z.

Bayesian Inference

Bayes' rule requires that the relationship between $\boldsymbol{x}$ and z be encoded as a joint probability or joint probability distribution $P(\boldsymbol{x}, z)$ for discrete and continuous variables, respectively. The chain rule of conditional probabilities can be used to expand a joint probability in two ways

$$P(\boldsymbol{x}, z) = P(\boldsymbol{x} \mid z)P(z) = P(z \mid \boldsymbol{x})P(\boldsymbol{x}). \tag{35.1}$$

Rearranging in terms of one of the conditionals, Bayes' rule is obtained

$$P(\boldsymbol{x} \mid z) = \frac{P(z \mid \boldsymbol{x})P(\boldsymbol{x})}{P(z)}. \tag{35.2}$$

The value of this result lies in the interpretation of the probabilities $P(\boldsymbol{x} \mid z)$, $P(z \mid \boldsymbol{x})$, and $P(\boldsymbol{x})$. Suppose it is necessary to determine the various likelihoods of different values of an unknown state $\boldsymbol{x}$. There may be prior beliefs about what values of $\boldsymbol{x}$ might be expected, encoded in the form of relative likelihoods in the *prior probability* $P(\boldsymbol{x})$. To obtain more information about the state $\boldsymbol{x}$ an observation z is made. These observations are modeled in the form of a conditional probability $P(z \mid \boldsymbol{x})$ that describes, for each fixed state $\boldsymbol{x}$, the probability that the observation z will be made, i. e., the probability of z given $\boldsymbol{x}$. The new likelihoods associated with the state $\boldsymbol{x}$ are computed from the product of the original prior information and the information gained by observation. This is encoded in the *posterior probability* $P(\boldsymbol{x} \mid z)$, which describes the likelihoods associated with $\boldsymbol{x}$ given the observation z. In this fusion process, the marginal probability $P(z)$ simply serves to normalize the posterior and is not generally computed. The marginal $P(z)$ plays an important role in model validation or data association as it provides a measure of how well the observation is predicted by the prior, because $P(z) = \int P(z \mid \boldsymbol{x})P(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x}$. The value of Bayes' rule is that it provides a principled means of combining observed information with prior beliefs about the state of the world.

Sensor Models and Multisensor Bayesian Inference

The conditional probability $P(z \mid \boldsymbol{x})$ serves the role of a *sensor model* and can be thought of in two ways. First, in building a sensor model, the probability is constructed by fixing the value of $\boldsymbol{x} = x$ and then asking what probability density $P(z \mid \boldsymbol{x} = x)$ on z results. Conversely, when this sensor model is used and observations are made, $z = z$ is fixed and a *likelihood function* $P(z = z \mid \boldsymbol{x})$ on $\boldsymbol{x}$ is inferred. The likelihood function, while not strictly a probability density, models the relative likelihood that different values of $\boldsymbol{x}$ gave rise to the observed value of z. The product of this likelihood with the prior, both defined on $\boldsymbol{x}$, gives the posterior or observation update $P(\boldsymbol{x} \mid z)$. In a practical implementation of (35.2), $P(z \mid \boldsymbol{x})$ is constructed as a function of both variables (or a matrix in discrete form). For each fixed value of $\boldsymbol{x}$, a probability density on z is defined. Therefore, as $\boldsymbol{x}$ varies, a family of likelihoods on z is created.

The multisensor form of Bayes' rule requires conditional independence

$$\begin{aligned} P(z_1, \cdots, z_n \mid \boldsymbol{x}) &= P(z_1 \mid \boldsymbol{x}) \cdots P(z_n \mid \boldsymbol{x}) \\ &= \prod_{i=1}^{n} P(z_i \mid \boldsymbol{x})\,, \end{aligned} \tag{35.3}$$

so that

$$P(\boldsymbol{x} \mid \boldsymbol{Z}^n) = CP(\boldsymbol{x}) \prod_{i=1}^{n} P(z_i \mid \boldsymbol{x})\,, \tag{35.4}$$

where C is a normalizing constant. Equation 35.4 is known as the *independent likelihood pool* [35.2]. This states that the posterior probability on $\boldsymbol{x}$ given all observations $\boldsymbol{Z}^n$, is simply proportional to the product of prior probability and individual likelihoods from each information source.

Avidbots Ex1016
Page 902 of 2154

The recursive form of Bayes' rule is

$$P(\boldsymbol{x} \mid \boldsymbol{Z}^k) = \frac{P(z_k \mid \boldsymbol{x}) P(\boldsymbol{x} \mid \boldsymbol{Z}^{k-1})}{P\left(z_k \mid \boldsymbol{Z}^{k-1}\right)} . \tag{35.5}$$

The advantage of (35.5) is that we need to compute and store only the posterior density $P(\boldsymbol{x} \mid \boldsymbol{Z}^{k-1})$, which contains a complete summary of all past information. When the next piece of information $P(z_k \mid \boldsymbol{x})$ arrives, the previous posterior takes on the role of the current prior and the product of the two becomes, when normalized, the new posterior.

Bayesian Filtering

Filtering is concerned with the sequential process of maintaining a probabilistic model for a state which evolves over time and which is periodically observed by a sensor. Filtering forms the basis for many problems in tracking and navigation. The general filtering problem can be formulated in Bayesian form. This is significant because it provides a common representation for a range of discrete and continuous data fusion problems without recourse to specific target or observation models.

Define $\boldsymbol{x}_t$ as the value of a state of interest at time t. This may, for example, describe a feature to be tracked, the state of a process being monitored, or the location of a platform for which navigation data is required. For convenience, and without loss of generality, time is defined at discrete (asynchronous) times $t_k \triangleq k$. At a time instant k, the following quantities are defined:

- $\boldsymbol{x}_k$: The state vector to be estimated at time k.
- $\boldsymbol{u}_k$: A control vector, assumed known, and applied at time $k-1$ to drive the state from $\boldsymbol{x}_{k-1}$ to $\boldsymbol{x}_k$ at time k.
- z_k: An observation taken of the state $\boldsymbol{x}_k$ at time k.

In addition, the following sets are also defined:

- The history of states: $\boldsymbol{X}^k = \{\boldsymbol{x}_0, \boldsymbol{x}_1, \cdots, \boldsymbol{x}_k\} = \{\boldsymbol{X}^{k-1}, \boldsymbol{x}_k\}$.
- The history of control inputs: $\boldsymbol{U}^k = \{\boldsymbol{u}_1, \boldsymbol{u}_2, \cdots, \boldsymbol{u}_k\} = \{\boldsymbol{U}^{k-1}, \boldsymbol{u}_k\}$.
- The history of state observations: $\boldsymbol{Z}^k = \{z_1, z_2, \cdots, z_k\} = \{\boldsymbol{Z}^{k-1}, z_k\}$.

In probabilistic form, the general data fusion problem is to find the posterior density

$$P\left(\boldsymbol{x}_k \mid \boldsymbol{Z}^k, \boldsymbol{U}^k, \boldsymbol{x}_0\right) \tag{35.6}$$

for all times k given the recorded observations and control inputs up to and including time k together (possibly) with knowledge of the initial state $\boldsymbol{x}_0$. Bayes' rule can be used to write (35.6) in terms of a sensor model $P(z_k \mid \boldsymbol{x}_k)$ and a predicted probability density $P(\boldsymbol{x}_k \mid \boldsymbol{Z}^{k-1}, \boldsymbol{U}^k, \boldsymbol{x}_0)$ based on observations up to time $k-1$ as

$$P\left(\boldsymbol{x}_k \mid \boldsymbol{Z}^k, \boldsymbol{U}^k, \boldsymbol{x}_0\right) = \frac{P(z_k \mid \boldsymbol{x}_k) P\left(\boldsymbol{x}_k \mid \boldsymbol{Z}^{k-1}, \boldsymbol{U}^k, \boldsymbol{x}_0\right)}{P\left(z_k \mid \boldsymbol{Z}^{k-1}, \boldsymbol{U}^k\right)} . \tag{35.7}$$

The denominator in (35.7) is independent of the state and following (35.4) can be set to some normalizing constant C. The sensor model makes use of the conditional independence assumption from (35.3).

The total probability theorem can be used to rewrite the second term in the numerator of (35.7) in terms of the state transition model and the joint posterior from time step $k-1$ as

$$\begin{aligned} & P\left(\boldsymbol{x}_k \mid \boldsymbol{Z}^{k-1}, \boldsymbol{U}^k, \boldsymbol{x}_0\right) \\ &= \int P\left(\boldsymbol{x}_k, \boldsymbol{x}_{k-1} \mid \boldsymbol{Z}^{k-1}, \boldsymbol{U}^k, \boldsymbol{x}_0\right) \mathrm{d}\boldsymbol{x}_{k-1} \\ &= \int P\left(\boldsymbol{x}_k \mid \boldsymbol{x}_{k-1}, \boldsymbol{Z}^{k-1}, \boldsymbol{U}^k, \boldsymbol{x}_0\right) \\ &\quad \times P\left(\boldsymbol{x}_{k-1} \mid \boldsymbol{Z}^{k-1}, \boldsymbol{U}^k, \boldsymbol{x}_0\right) \mathrm{d}\boldsymbol{x}_{k-1} \\ &= \int P(\boldsymbol{x}_k \mid \boldsymbol{x}_{k-1}, \boldsymbol{u}_k) \\ &\quad \times P\left(\boldsymbol{x}_{k-1} \mid \boldsymbol{Z}^{k-1}, \boldsymbol{U}^{k-1}, \boldsymbol{x}_0\right) \mathrm{d}\boldsymbol{x}_{k-1} , \end{aligned} \tag{35.8}$$

where the last equality implies that the future state depends only on the current state and the control exerted at this time. The state transition model is described in terms of a probability distribution in the form $P(\boldsymbol{x}_k \mid \boldsymbol{x}_{k-1}, \boldsymbol{u}_k)$. That is, the state transition may reasonably be assumed to be a Markov process in which the next state $\boldsymbol{x}_k$ depends only on the immediately proceeding state $\boldsymbol{x}_{k-1}$ and the applied control $\boldsymbol{u}_k$, and is independent of both the observations and preceding states.

Equations (35.7) and (35.8) define a recursive solution to (35.6). Equation (35.8) is the *time update* or prediction step for the full Bayes data fusion algorithm. A graphical description of this equation is shown in Fig. 35.1. Equation (35.7) is the *observation update* step for the full Bayes data fusion algorithm. A graphical description of this equation is shown in Fig. 35.2. The Kalman filter, grid-based methods, and sequential Monte Carlo methods, to be described, are specific implementations of these general equations.

35.1.2 Probabilistic Grids

Probabilistic grids are conceptually the simplest way of implementing Bayesian data fusion methods. They can

Avidbots Ex1016
Page 903 of 2154

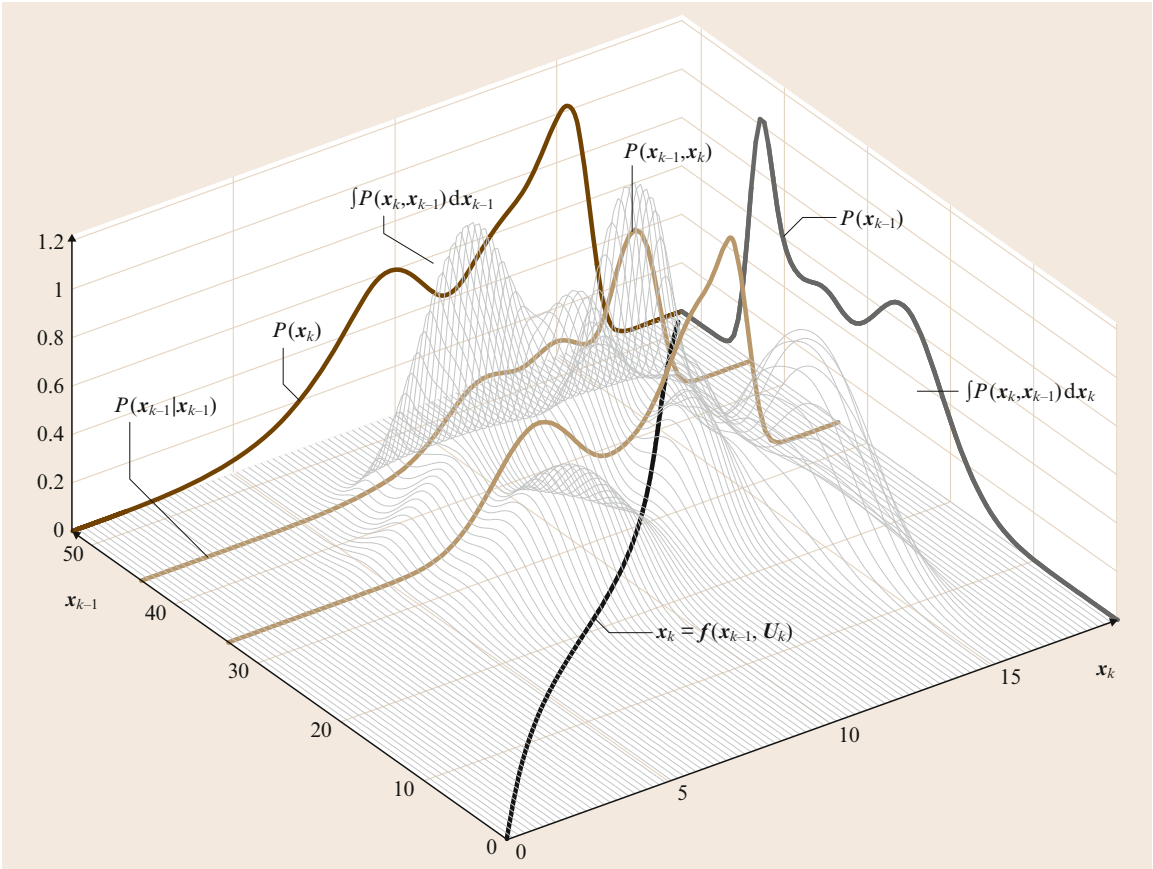


Fig. 35.1 Time update step for the full Bayes filter. At a time $k-1$, knowledge of the state $\boldsymbol{x}_{k-1}$ is summarized in a probability distribution $P(\boldsymbol{x}_{k-1})$. A vehicle model, in the form of a conditional probability density $P(\boldsymbol{x}_k \mid \boldsymbol{x}_{k-1})$, then describes the stochastic transition of the vehicle from a state $\boldsymbol{x}_{k-1}$ at a time $k-1$ to a state $\boldsymbol{x}_k$ at a time k. Functionally, this state transition may be related to an underlying kinematic state model in the form $\boldsymbol{x}_k = \boldsymbol{f}(\boldsymbol{x}_{k-1}, \boldsymbol{u}_k)$. The figure shows two typical conditional probability distributions $P(\boldsymbol{x}_k \mid \boldsymbol{x}_{k-1})$ on the state $\boldsymbol{x}_k$ given fixed values of $\boldsymbol{x}_{k-1}$. The product of this conditional distribution with the marginal distribution $P(\boldsymbol{x}_{k-1})$, describing the prior likelihood of values of $\boldsymbol{x}_k$, gives the the joint distribution $P(\boldsymbol{x}_k, \boldsymbol{x}_{k-1})$ shown as the surface in the figure. The total marginal density $P(\boldsymbol{x}_k)$ describes knowledge of $\boldsymbol{x}_k$ after state transition has occurred. The marginal density $P(\boldsymbol{x}_k)$ is obtained by integrating (projecting) the joint distribution $P(\boldsymbol{x}_k, \boldsymbol{x}_{k-1})$ over all $\boldsymbol{x}_{k-1}$. Equivalently, using the total probability theorem, the marginal density can be obtained by integrating (summing) all conditional densities $P(\boldsymbol{x}_k \mid \boldsymbol{x}_{k-1})$ weighted by the prior probability $P(\boldsymbol{x}_{k-1})$ of each $\boldsymbol{x}_{k-1}$. The process can equally be run in reverse (a retroverse motion model) to obtain $P(\boldsymbol{x}_{k-1})$ from $P(\boldsymbol{x}_k)$ given a model $P(\boldsymbol{x}_{k-1} \mid \boldsymbol{x}_k)$

be applied both to problems in mapping [35.3, 4] and tracking [35.5].

In mapping applications, the environment of interest is divided into a grid of equally sized spatial cells. Each cell is indexed and labeled with a property, thus the state $\boldsymbol{x}_{ij}$ may describe a two-dimensional world indexed by ij and having the property x. Interest is focused on maintaining a probability distribution on possible state values $P(\boldsymbol{x}_{ij})$ *at each grid cell*. Typically, in navigation and mapping problems, the property of interest has only two values O and E, *occupied* and *empty*, respectively, and it is then usual to assume that $P(\boldsymbol{x}_{ij} = O) = 1 - P(\boldsymbol{x}_{ij} = E)$. However, there is no particular constraint on the property encoded by the state $\boldsymbol{x}_{ij}$ which could have many values (green, red, blue, for example) and indeed be continuous (the temperature at a cell, for example).

Once the state has been defined, Bayesian methods require that a sensor model or likelihood function for the sensor be established. In theory, this requires specification of a probability distribution $P(\boldsymbol{z} \mid \boldsymbol{x}_{ij} = x_{ij})$ mapping each possible grid state x_{ij} to a distribution on

Avidbots Ex1016
Page 904 of 2154

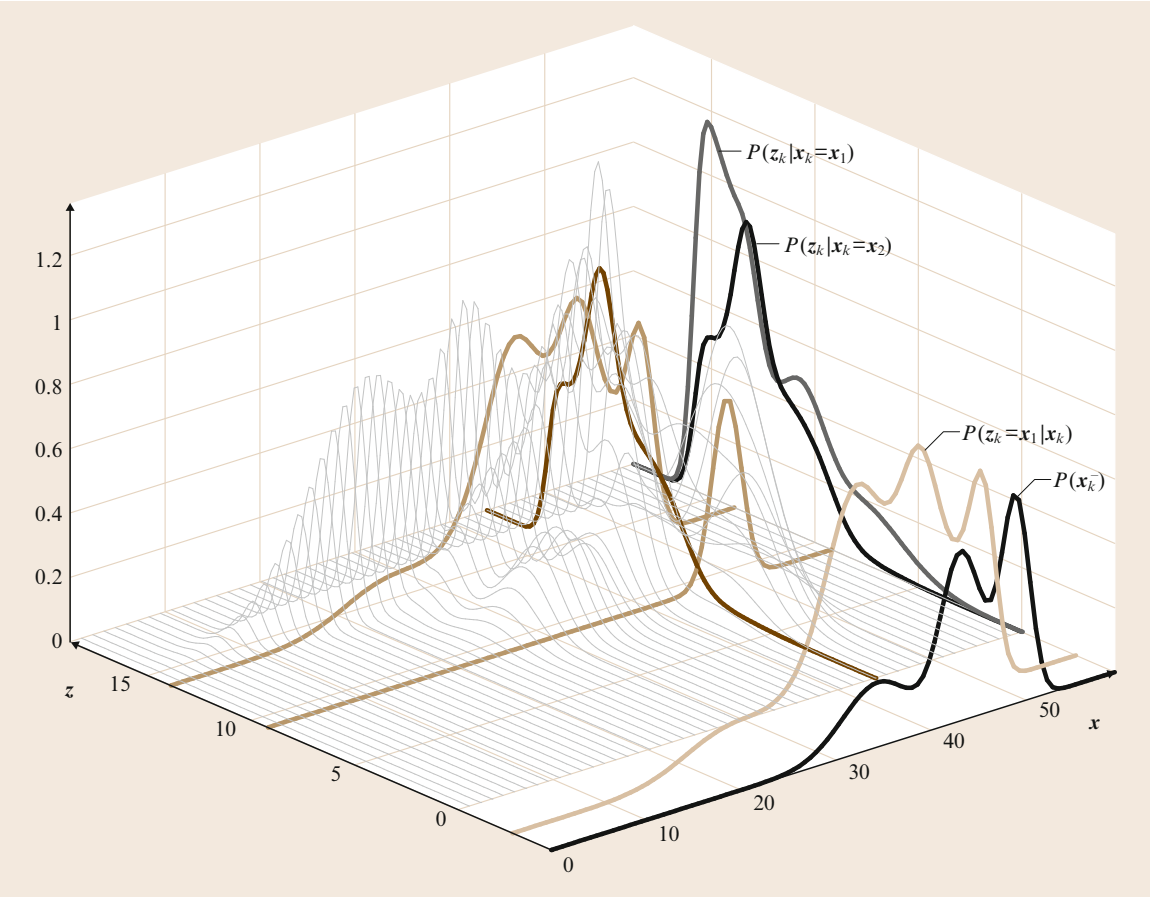


Fig. 35.2 Observation update for the full Bayes filter. Prior to observation, an observation model in the form of the conditional density $P(z_k \mid \boldsymbol{x}_k)$ is established. For a fixed value of $\boldsymbol{x}_k$, equal to $\boldsymbol{x}_1$ or $\boldsymbol{x}_2$ for example, a density function $P(z_k \mid \boldsymbol{x}_k = \boldsymbol{x}_1)$ or $P(z_k \mid \boldsymbol{x}_k = \boldsymbol{x}_2)$ is defined describing the likelihood of making the observation z_k. Together the density $P(z_k \mid \boldsymbol{x}_k)$ is then a function of both z_k and $\boldsymbol{x}_k$. This conditional density then defines the observation model. Now, in operation, a specific observation $z_k = \boldsymbol{x}_1$ is made and the resulting distribution $P(z_k = \boldsymbol{x}_1 \mid \boldsymbol{x}_k)$ defines a density function (now termed the likelihood function) on $\boldsymbol{x}_k$. This density is then multiplied by the prior density $P(\boldsymbol{x}_k^-)$ and normalized to obtain the posterior distribution $P(\boldsymbol{x}_k \mid z_k)$ describing knowledge in the state after observation

observations. Practically, however, this is implemented simply as another *observation* grid so that for a specific observation $\boldsymbol{z} = z$ (taken from a specific location), a grid of likelihoods on the states $\boldsymbol{x}_{ij}$ is produced in the form $P(\boldsymbol{z} = z \mid \boldsymbol{x}_{ij}) = \Lambda(\boldsymbol{x}_{ij})$. It is then trivial to apply Bayes' rule to update the property value at each grid cell as

$$P^+(\boldsymbol{x}_{ij}) = C\Lambda(\boldsymbol{x}_{ij})P(\boldsymbol{x}_{ij}) \,, \quad \forall i, j \,, \tag{35.9}$$

where C us a normalizing constant obtained by summing posterior probabilities to one *at node ij only*. Computationally, this is a simple pointwise multiplication of two grids. Some care needs to be taken that the two grids appropriately overlap and align with each other at the right scale. In some instances it is also valuable to encode the fact that spatially adjacent cells will influence each other; that is, if we knew the value of the property (occupancy, temperature, for example) at ij we will have some belief also of the value of this property at adjacent nodes $i+1, j$, $i, j+1$, etc. Different sensors and the fusion of different sensor outputs is accommodated simply by building appropriate sensor models $\Lambda(\boldsymbol{x}_{ij})$.

Grids can also be used for tracking and self-tracking (localization). The state $\boldsymbol{x}_{ij}$ in this case is the location of the entity being tracked. This is a qualitatively different definition of state from that used in mapping. The probability $P(\boldsymbol{x}_{ij})$ must now be interpreted as the probability that the object being tracked occupies the grid cell ij. In the case of mapping, the sum of property probabilities at each grid cell is one, whereas in the case of tracking, the

Avidbots Ex1016
Page 905 of 2154

sum of location probabilities over *the whole grid* must sum to one. Otherwise, the procedure for updating is very similar. An observation grid is constructed which, when instantiated with an observation value, provides a location likelihood grid

$$P(\boldsymbol{z} = z \mid \boldsymbol{x}_{ij}) = \Lambda(\boldsymbol{x}_{ij}) \, .$$

Bayes' rule is then applied to update the location probability at each grid cell in the same form as (35.9) except that now the normalization constant C is obtained by summing posterior probabilities over *all* ij grid cells. This can become computationally expensive, especially if the grid has three or more dimensions. One major advantage of grid-based tracking is that it is easy to incorporate quite complex prior information. For example, if it is known that the object being tracked is on a road, then the probability location values for all off-road grid cells can simply be set to zero.

Grid-based fusion is appropriate to situations where the domain size and dimension are modest. In such cases, grid-based methods provide straightforward and effective fusion algorithms. Grid-based methods can be extended in a number of ways; to hierarchical (quad-tree) grids, or to irregular (triangular, pentagonal) grids. These can help reduce computation in larger spaces. Monte Carlo and particle filtering methods (Sect. 35.1.4) may be considered as grid-based methods, where the grid cells themselves are sample of the underlying probability density for the state.

35.1.3 The Kalman Filter

The Kalman filter is a recursive linear estimator that successively calculates an estimate for a continuous valued state, that evolves over time, on the basis of periodic observations of the state. The Kalman filter employs an explicit statistical model of how the parameter of interest $\boldsymbol{x}(t)$ evolves over time and an explicit statistical model of how the observations $\boldsymbol{z}(t)$ that are made are related to this parameter. The gains employed in a Kalman filter are chosen to ensure that, with certain assumptions about the observation and process models used, the resulting estimate $\hat{\boldsymbol{x}}(t)$ minimizes mean-squared error and is thus the conditional mean $\hat{\boldsymbol{x}}(t) = \mathrm{E}[\boldsymbol{x}(t) \mid \boldsymbol{Z}^t]$: an average, rather than a most likely value.

The Kalman filter has a number of features which make it ideally suited to dealing with complex multisensor estimation and data fusion problems. In particular, the explicit description of process and observations allows a wide variety of different sensor models to be incorporated within the basic algorithm. In addition, the consistent use of statistical measures of uncertainty makes it possible to quantitatively evaluate the role each sensor plays in overall system performance. Further, the linear recursive nature of the algorithm ensures that its application is simple and efficient. For these reasons, the Kalman filter has found widespread application in many different data fusion problems [35.6–9].

In robotics, the Kalman filter is most suited to problems in tracking, localization, and navigation, and less so to problems in mapping. This is because the algorithm works best with well-defined state descriptions (positions, velocities, for example), and for states where observation and time-propagation models are also well understood.

Observation and Transition Models

The Kalman filter may be considered a specific instance of the recursive Bayesian filter of (35.7) and (35.8) for the case where the probability densities on states are Gaussian. The starting point for the Kalman Filter algorithm is to define a model for the states to be estimated in the standard state-space form

$$\dot{\boldsymbol{x}}(t) = \mathbf{F}(t)\boldsymbol{x}(t) + \mathbf{B}(t)\boldsymbol{u}(t) + \mathbf{G}(t)\boldsymbol{v}(t) \, , \qquad (35.10)$$

where $\boldsymbol{x}(t)$ is the state vector of interest, $\boldsymbol{u}(t)$ is a known control input, $\boldsymbol{v}(t)$ is a random variable describing uncertainty in the evolution of the state, and $\mathbf{F}(t)$, $\mathbf{B}(t)$, and $\mathbf{G}(t)$ are matrices describing the contribution of states, controls, and noise to state transition [35.7]. An observation (output) model is also defined in standard state-space form

$$\boldsymbol{z}(t) = \mathbf{H}(t)\boldsymbol{x}(t) + \mathbf{D}(t)\boldsymbol{w}(t) \, , \qquad (35.11)$$

where $\boldsymbol{z}(t)$ is the observation vector, $\boldsymbol{w}(t)$ is a random variable describing uncertainty in the observation, and where $\mathbf{H}(t)$ and $\mathbf{D}(t)$ are matrices describing the contribution of state and noise to the observation.

These equations define the evolution of a continuous-time system with continuous observations being made of the state. However, the Kalman Filter is almost always implemented in discrete time $t_k = k$. It is straightforward [35.8] to obtain a discrete-time version of (35.10) and (35.11) in the form

$$\boldsymbol{x}(k) = \mathbf{F}(k)\boldsymbol{x}(k-1) + \mathbf{B}(k)\boldsymbol{u}(k) + \mathbf{G}(k)\boldsymbol{v}(k) \, , \qquad (35.12)$$

$$\boldsymbol{z}(k) = \mathbf{H}(k)\boldsymbol{x}(k) + \mathbf{D}(k)\boldsymbol{w}(k) \, . \qquad (35.13)$$

A basic assumption in the derivation of the Kalman filter is that the random sequences $\boldsymbol{v}(k)$ and $\boldsymbol{w}(k)$ describing process and observation noise are all Gaussian, temporally uncorrelated, and zero-mean

$$\mathrm{E}[\boldsymbol{v}(k)] = \mathrm{E}[\boldsymbol{w}(k)] = \mathbf{0} \, , \quad \forall k \, , \qquad (35.14)$$

Avidbots Ex1016
Page 906 of 2154

with known covariance

$$\mathrm{E}[\boldsymbol{v}(i)\boldsymbol{v}^{\mathrm{T}}(j)] = \delta_{ij}\mathbf{Q}(i)\,, \quad \mathrm{E}[\boldsymbol{w}(i)\boldsymbol{w}^{\mathrm{T}}(j)] = \delta_{ij}\mathbf{R}(i)\,. \tag{35.15}$$

It is also generally assumed that the process and observation noises are also uncorrelated

$$\mathrm{E}[\boldsymbol{v}(i)\boldsymbol{w}^{\mathrm{T}}(j)] = \mathbf{0}\,, \quad \forall i,j\,. \tag{35.16}$$

These are equivalent to a Markov property requiring observations and successive states to be conditionally independent. If the sequences $\boldsymbol{v}(k)$ and $\boldsymbol{w}(k)$ are temporally correlated, a shaping filter can be used to whiten the observations, again making the assumptions required for the Kalman filter valid [35.8]. If the process and observation noise sequences are correlated, then this correlation can also be accounted for in the Kalman filter algorithm [35.10]. If the sequence is not Gaussian, but is symmetric with finite moments, then the Kalman filter will still produce good estimates. If however, the sequence has a distribution which is skewed or otherwise pathological, results produced by the Kalman filter will be misleading and there will be a good case for using a more-sophisticated Bayesian filter [35.5].

Filtering Algorithm

The Kalman filter algorithm produces estimates that minimize mean-squared estimation error conditioned on a given observation sequence and so is the conditional mean

$$\hat{\boldsymbol{x}}(i\,|\,j) \triangleq \mathrm{E}[\boldsymbol{x}(i)\,|\,z(1),\cdots,z(j)] \triangleq \mathrm{E}[\boldsymbol{x}(i)\,|\,\boldsymbol{Z}^j]\,. \tag{35.17}$$

The estimate variance is defined as the mean-squared error in this estimate

$$\boldsymbol{P}(i\,|\,j) \triangleq \mathrm{E}\{[\boldsymbol{x}(i)-\hat{\boldsymbol{x}}(i\,|\,j)][\boldsymbol{x}(i)-\hat{\boldsymbol{x}}(i\,|\,j)]^T\,|\,\boldsymbol{Z}^j\}\,. \tag{35.18}$$

The estimate of the state at a time k given all information up to time k is written $\hat{\boldsymbol{x}}(k\,|\,k)$. The estimate of the state at a time k given only information up to time $k-1$ is called a one-step-ahead prediction (or just a prediction) and is written $\hat{\boldsymbol{x}}(k\,|\,k-1)$.

The Kalman filter algorithm is now stated without proof. Detailed derivations can be found in many books on the subject, for example [35.7, 8]. The state is assumed to evolve in time according to (35.12). Observations of this state are made at regular time intervals according to (35.13). The noise processes entering the system are assumed to obey (35.14–35.16). It is also assumed that an estimate $\hat{\boldsymbol{x}}(k-1\,|\,k-1)$ of the state $\boldsymbol{x}(k-1)$ at time $k-1$ based on all observations made up to and including time $k-1$ is available, and that this estimate is equal to the conditional mean of the true state $\boldsymbol{x}(k-1)$ conditioned on these observations. The conditional variance $\boldsymbol{P}(k-1\,|\,k-1)$ in this estimate is also assumed known. The Kalman filter then proceeds recursively in two stages (Fig. 35.3).

Prediction. A prediction $\hat{\boldsymbol{x}}(k\,|\,k-1)$ of the state at time k and its covariance $\boldsymbol{P}(k\,|\,k-1)$ is computed according to

$$\hat{\boldsymbol{x}}(k\,|\,k-1) = \mathbf{F}(k)\hat{\boldsymbol{x}}(k-1\,|\,k-1) + \mathbf{B}(k)\boldsymbol{u}(k)\,, \tag{35.19}$$

$$\boldsymbol{P}(k\,|\,k-1) = \mathbf{F}(k)\boldsymbol{P}(k-1\,|\,k-1)\mathbf{F}^{\mathrm{T}}(k) + \mathbf{G}(k)\mathbf{Q}(k)\mathbf{G}^{\mathrm{T}}(k)\,. \tag{35.20}$$

Update. At time k an observation $\boldsymbol{z}(k)$ is made and the updated estimate $\hat{\boldsymbol{x}}(k\,|\,k)$ of the state $\boldsymbol{x}(k)$, together with the updated estimate covariance $\boldsymbol{P}(k\,|\,k)$, is computed from the state prediction and observation according to

$$\hat{\boldsymbol{x}}(k\,|\,k) = \hat{\boldsymbol{x}}(k\,|\,k-1) + \mathbf{W}(k)[\boldsymbol{z}(k) - \mathbf{H}(k)\hat{\boldsymbol{x}}(k\,|\,k-1)]\,, \tag{35.21}$$

$$\boldsymbol{P}(k\,|\,k) = \boldsymbol{P}(k\,|\,k-1) - \mathbf{W}(k)\mathbf{S}(k)\mathbf{W}^{\mathrm{T}}(k)\,, \tag{35.22}$$

where the gain matrix $\mathbf{W}(k)$ is given by

$$\mathbf{W}(k) = \boldsymbol{P}(k\,|\,k-1)\mathbf{H}(k)\mathbf{S}^{-1}(k)\,, \tag{35.23}$$

where

$$\mathbf{S}(k) = \mathbf{R}(k) + \mathbf{H}(k)\boldsymbol{P}(k\,|\,k-1)\mathbf{H}(k) \tag{35.24}$$

is the innovation covariance. The difference between the observation $\boldsymbol{z}(k)$ and the predicted observation $\mathbf{H}(k)\hat{\boldsymbol{x}}(k\,|\,k-1)$ is termed the *innovation* or residual $\boldsymbol{v}(k)$

$$\boldsymbol{v}(k) = \boldsymbol{z}(k) - \mathbf{H}(k)\hat{\boldsymbol{x}}(k\,|\,k-1)\,. \tag{35.25}$$

The innovation is an important measure of the deviation between the filter estimates and the observation sequence. Indeed, because the true states are not usually available for comparison with the estimated states, the innovation is often the only measure of how well the estimator is performing. The innovation is particularly important in data association.

The Extended Kalman Filter

The extended Kalman filter (EKF) is a form of the Kalman filter that can be employed when the state model and/or the observation model are nonlinear. The EKF is briefly described in this section.

Avidbots Ex1016
Page 907 of 2154

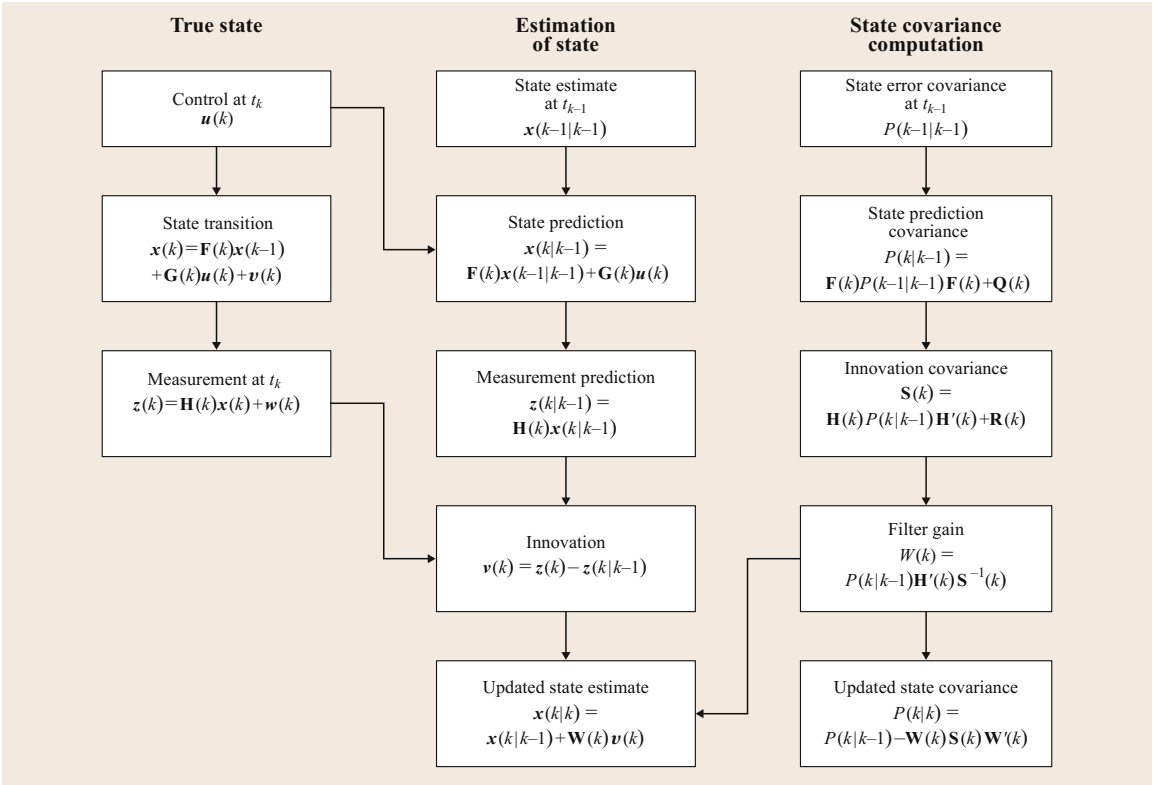


Fig. 35.3 Block diagram of the Kalman filter cycle (after *Bar-Shalom* and *Fortmann* [35.7])

The state models considered by the EKF are described in state-space notation by a first-order nonlinear vector differential equation or state model of the form

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}[\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{v}(t), t] \,, \tag{35.26}$$

where $\boldsymbol{f}[\cdot,\cdot,\cdot,\cdot]$ is now a general nonlinear mapping of state and control input to state transition. The observation models considered by the EKF are described in state-space notation by a nonlinear vector function in the form

$$\boldsymbol{z}(t) = \boldsymbol{h}[\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{w}(t), t] \,, \tag{35.27}$$

where $\boldsymbol{h}[\cdot,\cdot,\cdot,\cdot]$ is now a general nonlinear mapping of state and control input to observations.

The EKF, like the Kalman filter, is almost always implemented in discrete time. By integration and with appropriate identification of discrete time states and observations, the state model is written

$$\boldsymbol{x}(k) = \boldsymbol{f}[\boldsymbol{x}(k-1), \boldsymbol{u}(k), \boldsymbol{v}(k), k] \,, \tag{35.28}$$

and the observation model as

$$\boldsymbol{z}(k) = \boldsymbol{h}[\boldsymbol{x}(k), \boldsymbol{w}(k)] \,. \tag{35.29}$$

Like the Kalman filter, it is assumed that the noises $\boldsymbol{v}(k)$ and $\boldsymbol{w}(k)$ are all Gaussian, temporally uncorrelated, and zero-mean with known variance as defined in (35.14–35.16). The EKF aims to minimize the mean-squared error and thereby compute an approximation to the conditional mean. It is assumed therefore that an estimate of the state at time $k-1$ is available which is approximately equal to the conditional mean, $\hat{\boldsymbol{x}}(k-1 \mid k-1) \approx \mathrm{E}[\boldsymbol{x}(k-1) \mid Z^{k-1}]$. The EKF algorithm will now be stated without proof. Detailed derivations may be found in any number of books on the subject. The principle stages in the derivation of the EKF follow directly from those of the linear Kalman filter with the additional step that the process and observation models are linearized as a Taylor series about the estimate and prediction, respectively. The algorithm has two stages: prediction and update.

Prediction. A prediction $\hat{\boldsymbol{x}}(k \mid k-1)$ of the state at time k and its covariance $\boldsymbol{P}(k \mid k-1)$ is computed according to

$$\hat{\boldsymbol{x}}(k \mid k-1) = \boldsymbol{f}[\hat{\boldsymbol{x}}(k-1 \mid k-1), \boldsymbol{u}(k)] \,, \tag{35.30}$$

$$\boldsymbol{P}(k \mid k-1) = \nabla \boldsymbol{f}_{\boldsymbol{x}}(k) \boldsymbol{P}(k-1 \mid k-1) \nabla^{\mathrm{T}} \boldsymbol{f}_{\boldsymbol{x}}(k) + \nabla \boldsymbol{f}_{\boldsymbol{v}}(k) \mathbf{Q}(k) \nabla^{\mathrm{T}} \boldsymbol{f}_{\boldsymbol{v}}(k) \,. \tag{35.31}$$

Avidbots Ex1016
Page 908 of 2154

Update. At time k an observation $\boldsymbol{z}(k)$ is made and the updated estimate $\hat{\boldsymbol{x}}(k \mid k)$ of the state $\boldsymbol{x}(k)$, together with the updated estimate covariance $\boldsymbol{P}(k \mid k)$ is computed from the state prediction and observation according to

$$\hat{\boldsymbol{x}}(k \mid k) = \hat{\boldsymbol{x}}(k \mid k-1) + \mathbf{W}(k)\{\boldsymbol{z}(k) - \boldsymbol{h}[\hat{\boldsymbol{x}}(k \mid k-1)]\} , \quad (35.32)$$

$$\boldsymbol{P}(k \mid k) = \boldsymbol{P}(k \mid k-1) - \mathbf{W}(k)\mathbf{S}(k)\mathbf{W}^{\mathrm{T}}(k) , \quad (35.33)$$

where

$$\mathbf{W}(k) = \boldsymbol{P}(k \mid k-1)\nabla^{\mathrm{T}}\boldsymbol{h}_{\boldsymbol{x}}(k)\mathbf{S}^{-1}(k) , \quad (35.34)$$

and

$$\mathbf{S}(k) = \nabla\boldsymbol{h}_{w}(k)\mathbf{R}(k)\nabla^{\mathrm{T}}\boldsymbol{h}_{w}(k) + \nabla\boldsymbol{h}_{\boldsymbol{x}}(k)\boldsymbol{P}(k \mid k-1)\nabla^{\mathrm{T}}\boldsymbol{h}_{\boldsymbol{x}}(k) , \quad (35.35)$$

and where the Jacobian $\nabla\boldsymbol{f}_{\cdot}(k)$ is evaluated at $\boldsymbol{x}(k-1) = \hat{\boldsymbol{x}}(k-1 \mid k-1)$ and $\nabla\boldsymbol{h}_{\cdot}(k)$ is evaluated at $\boldsymbol{x}(k) = \hat{\boldsymbol{x}}(k \mid k-1)$.

A comparison of (35.19–35.24) with (35.30–35.35) makes it clear that the EKF algorithm is very similar to the linear Kalman filter algorithm, with the substitutions $\mathbf{F}(k) \to \nabla\boldsymbol{f}_{\boldsymbol{x}}(k)$ and $\mathbf{H}(k) \to \nabla\boldsymbol{h}_{\boldsymbol{x}}(k)$ being made in the equations for the variance and gain propagation. Thus, the EKF is, in effect, a linear estimator for a state error which is described by a *linear* equation and which is being observed according to a *linear* equation of the form of (35.13).

The EKF works in much the same way as the linear Kalman filter with some notable caveats:

- The Jacobians $\nabla\boldsymbol{f}_{\boldsymbol{x}}(k)$ and $\nabla\boldsymbol{h}_{\boldsymbol{x}}(k)$ are typically not constant, being functions of both state and time step. This means that unlike the linear filter, the covariances and gain matrix must be computed online as estimates and predictions are made available, and will not in general tend to constant values. This significantly increases the amount of computation which must be performed online by the algorithm.
- As the linearized model is derived by perturbing the true state and observation models around a predicted or nominal trajectory, great care must be taken to ensure that these predictions are always close enough to the true state that second order terms in the linearization are indeed insignificant. If the nominal trajectory is too far away from the true trajectory then the true covariance will be much larger than the estimated covariance and the filter will become poorly matched. In extreme cases the filter may also become unstable.
- The EKF employs a linearized model that must be computed from an approximate knowledge of the state. Unlike the linear algorithm this means that the filter must be accurately initialized at the start of operation to ensure that the linearized models obtained are valid. If this is not done, the estimates computed by the filter will simply be meaningless.

The Information Filter

The information filter is mathematically equivalent to a Kalman filter. However, rather than generating state estimates $\hat{\boldsymbol{x}}(i \mid j)$ and covariances $\boldsymbol{P}(i \mid j)$ it uses information state variables $\hat{\boldsymbol{y}}(i \mid j)$ and information matrices $\mathbf{Y}(i \mid j)$, which are related to each other through the relationships

$$\hat{\boldsymbol{y}}(i \mid j) = \boldsymbol{P}^{-1}(i \mid j)\hat{\boldsymbol{x}}(i \mid j) ,$$
$$\mathbf{Y}(i \mid j) = \boldsymbol{P}^{-1}(i \mid j) . \quad (35.36)$$

The information filter has the same prediction–update structure as the Kalman filter.

Prediction. A prediction $\hat{\boldsymbol{y}}(k \mid k-1)$ of the information state at time k and its information matrix $\mathbf{Y}(k \mid k-1)$ is computed according to (Joseph form [35.8])

$$\hat{\boldsymbol{y}}(k \mid k-1) = (\mathbf{1} - \boldsymbol{\Omega}\mathbf{G}^{\mathrm{T}})\mathbf{F}^{-\mathrm{T}}\hat{\boldsymbol{y}}(k-1 \mid k-1) + \mathbf{Y}(k \mid k-1)\mathbf{B}\boldsymbol{u}(k) , \quad (35.37)$$

$$\mathbf{Y}(k \mid k-1) = \mathbf{M}(k) - \boldsymbol{\Omega}\boldsymbol{\Sigma}\boldsymbol{\Omega}^{\mathrm{T}} , \quad (35.38)$$

respectively, where

$$\mathbf{M}(k) = \mathbf{F}^{-\mathrm{T}}\mathbf{Y}(k-1 \mid k-1)\mathbf{F}^{-1} ,$$
$$\boldsymbol{\Sigma} = \mathbf{G}^{\mathrm{T}}\mathbf{M}(k)\mathbf{G} + \mathbf{Q}^{-1} ,$$

and

$$\boldsymbol{\Omega} = \mathbf{M}(t_k)\mathbf{G}\boldsymbol{\Sigma}^{-1} .$$

It should be noted that $\boldsymbol{\Sigma}$, whose inverse is required to compute $\boldsymbol{\Omega}$, has the same dimensionality as the process driving noise, which is normally considerably smaller than the state dimension. Further, the matrix $\mathbf{F}^{-1}$ is the state-transition matrix evaluated backwards in time and so must always exist.

Update. At time k an observation $\boldsymbol{z}(k)$ is made and the updated information state estimate $\hat{\boldsymbol{y}}(k \mid k)$ together with the updated information matrix $\mathbf{Y}(k \mid k)$ is computed from

$$\hat{\boldsymbol{y}}(k \mid k) = \hat{\boldsymbol{y}}(k \mid k-1) + \mathbf{H}(k)\mathbf{R}^{-1}(k)\boldsymbol{z}(k) , \quad (35.39)$$

$$\mathbf{Y}(k \mid k) = \mathbf{Y}(k \mid k-1) + \mathbf{H}(k)\mathbf{R}^{-1}(k)\mathbf{H}^{\mathrm{T}}(k) . \quad (35.40)$$

Avidbots Ex1016
Page 909 of 2154

We emphasize that (35.38) and (35.37) are mathematically identical to (35.19) and (35.20), and that (35.39) and (35.40) are mathematically identical to (35.21) and (35.22). It will be noted that there is a duality between information and state space forms [35.10]. This duality is evident from the fact that $\mathbf{\Omega}$ and $\mathbf{\Sigma}$ in the prediction stage of the information filter play an equivalent role to the gain matrix $\mathbf{W}$ and innovation covariance $\mathbf{S}$ in the update stage of the Kalman filter. Furthermore, the simple linear update step for the information filter is mirrored in the simple linear prediction step for the Kalman filter.

The main advantage of the information filter over the Kalman filter in data fusion problems is the relative simplicity of the update stage. For a system with n sensors, the fused information state update is *exactly* the linear sum of information contributions from all sensors as

$$\hat{\boldsymbol{y}}(k \mid k) = \hat{\boldsymbol{y}}(k \mid k-1) + \sum_{i=1}^{n} \mathbf{H}_i(k)\mathbf{R}_i^{-1}(k)\boldsymbol{z}_i(k) \,,$$
$$\mathbf{Y}(k \mid k) = \mathbf{Y}(k \mid k-1) + \sum_{i=1}^{n} \mathbf{H}_i(k)\mathbf{R}_i^{-1}(k)\mathbf{H}_i^{\mathrm{T}}(k) \,. \tag{35.41}$$

The reason such an expression exists in this form is that the information filter is essentially a log-likelihood expression of Bayes' rule, where products of likelihoods (35.4) are turned into sums. No such simple expression for multisensor updates exists for the Kalman filter. This property of the information filter has been exploited for data fusion in robotic networks [35.11, 12] and more recently in robot navigation and localization problems [35.1]. One substantial disadvantage of the information filter is the coding of nonlinear models, especially for the prediction step.

When to Use a Kalman or Information Filter

Kalman or information filters are appropriate to data fusion problems where the entity of interest is well defined by a continuous parametric state. This would include estimation of the position, attitude, and velocity of a robot or other object, or the tracking of a simple geometric feature such as a point, line or curve. Kalman and information filters are inappropriate for estimating properties such as spatial occupancy, discrete labels, or processes whose error characteristics are not easily parameterized.

35.1.4 Sequential Monte Carlo Methods

Monte Carlo (MC) filter methods describe probability distributions as a set of weighted samples of an underlying state space. MC filtering then uses these samples to simulate probabilistic inference usually through Bayes' rule. Many samples or simulations are performed. By studying the statistics of these samples as they progress through the inference process, a probabilistic picture of the process being simulated can be built up.

Representing Probability Distributions

In sequential Monte Carlo methods, probability distributions are described in terms of a set of support points (state-space values) $\boldsymbol{x}^i$, $i = 1, \cdots, N$, together with a corresponding set of normalized weights w^i, $i = 1, \cdots, N$, where $\sum_i w^i = 1$. The support points and weights can be used to define a probability density function in the form

$$P(\boldsymbol{x}) \approx \sum_{i=1}^{N} w^i \delta(\boldsymbol{x} - \boldsymbol{x}^i) \,. \tag{35.42}$$

A key question is how these support points and weights are selected to obtain a faithful representation of the probability density $P(\boldsymbol{x})$. The most general way of selecting support values is to use an *importance density* $q(\boldsymbol{x})$. The support values $\boldsymbol{x}^i$ are drawn as samples from this density; where the density has high probability, more support values are chosen, and where the density has low probability, few support support vectors are selected. The weights in (35.42) are then computed from

$$w^i \propto \frac{P(\boldsymbol{x}^i)}{q(\boldsymbol{x}^i)} \,. \tag{35.43}$$

Practically, a sample $\boldsymbol{x}^i$ is drawn from the importance distribution. The sample is then instantiated in the underlying probability distribution to yield the value $P(\boldsymbol{x} = \boldsymbol{x}^i)$. The ratio of the two probability values, appropriately normalized, then becomes the weight.

There are two instructive extremes of the importance sampling method:

1. At one extreme, the importance density could be taken to be a uniform distribution, so the support values $\boldsymbol{x}^i$ are uniformly distributed on the state space in a close approximation to a grid. The probabilities $q(\boldsymbol{x}^i)$ are also therefore equal. The weights computed from (35.43) are then simply proportional to the probabilities $w^i \propto P(\boldsymbol{x} = \boldsymbol{x}^i)$. The result is a model for the distribution which looks very like the regular grid model.
2. At the other extreme, we could choose an importance density equal to the probability model $q(\boldsymbol{x}) = P(\boldsymbol{x})$. Samples of the support values $\boldsymbol{x}^i$ are now drawn from this density. Where the density is high

Avidbots Ex1016
Page 910 of 2154

there will be many samples, where the density is low there will be few samples. However, if we substitute $q(\boldsymbol{x}^i) = P(\boldsymbol{x}^i)$ into (35.43), it is clear that the weights all become equal $w^i = \frac{1}{N}$. A set of samples with equal weights is known as a *particle* distribution.

It is, of course, possible to mix these two representations to describe a probability distribution both in terms of a set of weights and in terms of a set of support values. The complete set of samples and weights describing a probability distribution $\{\boldsymbol{x}^i, w^i\}_{i=1}^N$ is termed a *random measure*.

The Sequential Monte Carlo Method

Sequential Monte Carlo (SMC) filtering is a simulation of the recursive Bayes update equations using sample support values and weights to describe the underlying probability distributions.

The starting point is the recursive or sequential Bayes observation update given in (35.7) and (35.8). The SMC recursion begins with an a posterior probability density represented by a set of support values and weights $\{\boldsymbol{x}_{k-1}^i, w_{k-1|k-1}^i\}_{i=1}^{N_{k-1}}$ in the form

$$P\left(\boldsymbol{x}_{k-1} \mid \boldsymbol{Z}^{k-1}\right) = \sum_{i=1}^{N_{k-1}} w_{k-1}^i \delta\left(\boldsymbol{x}_{k-1} - \boldsymbol{x}_{k-1}^i\right) . \tag{35.44}$$

The prediction step requires that (35.44) is substituted into (35.8) where the joint density is marginalized. Practically however, this complex step is avoided by implicitly assuming that the importance density is exactly the transition model as

$$q_k\left(\boldsymbol{x}_k^i\right) = P\left(\boldsymbol{x}_k^i \mid \boldsymbol{x}_{k-1}^i\right) . \tag{35.45}$$

This allows new support values $\boldsymbol{x}_k^i$ to be drawn on the basis of old support values $\boldsymbol{x}_{k-1}^i$ while leaving the weights unchanged $w_k^i = w_{k-1}^i$. With this, the prediction becomes

$$P\left(\boldsymbol{x}_k \mid \boldsymbol{Z}^{k-1}\right) = \sum_{i=1}^{N_k} w_{k-1}^i \delta\left(\boldsymbol{x}_k - \boldsymbol{x}_k^i\right) . \tag{35.46}$$

The SMC observation update step is relatively straightforward. An observation model $P(\boldsymbol{z}_k \mid \boldsymbol{x}_k)$ is defined. This is a function on both variables, $\boldsymbol{z}_k$ and $\boldsymbol{x}_k$, and is a probability distribution on $\boldsymbol{z}_k$ (integrates to unity). When an observation or measurement is made, $\boldsymbol{z}_k = z_k$, the observation model becomes a function of state $\boldsymbol{x}_k$ only. If samples of the state are taken $\boldsymbol{x}_k = \boldsymbol{x}_k^i$, $i = 1 \cdots, N_k$, the observation model $P(\boldsymbol{z}_k = z_k \mid \boldsymbol{x}_k = \boldsymbol{x}_k^i)$ becomes a set of scalars describing the likelihood that the sample $\boldsymbol{x}_k^i$ could have given rise to the observation z_k. Substituting these likelihoods and (35.46) into (35.7) gives

$$P\left(\boldsymbol{x}_k \mid \boldsymbol{Z}^k\right) = C \sum_{i=1}^{N_k} w_{k-1}^i P\left(\boldsymbol{z}_k = z_k \mid \boldsymbol{x}_k = \boldsymbol{x}_k^i\right) \delta\left(\boldsymbol{x}_k - \boldsymbol{x}_k^i\right) . \tag{35.47}$$

This is normally implemented in the form of an updated set of normalized weights

$$w_k^i = \frac{w_{k-1}^i P\left(\boldsymbol{z}_k = z_k \mid \boldsymbol{x}_k = \boldsymbol{x}_k^i\right)}{\sum_{j=1}^{N_k} w_{k-1}^j P\left(\boldsymbol{z}_k = z_k \mid \boldsymbol{x}_k = \boldsymbol{x}_k^j\right)} , \tag{35.48}$$

and so

$$P\left(\boldsymbol{x}_k \mid \boldsymbol{Z}^k\right) = \sum_{i=1}^{N_k} w_k^i \delta\left(\boldsymbol{x}_k - \boldsymbol{x}_k^i\right) . \tag{35.49}$$

Note that the support values in (35.49) are the same as those in (35.46), only the weights have been changed by the observation update.

The implementation of the SMC method requires the enumeration of models for both the state transition $P(\boldsymbol{x}_k \mid \boldsymbol{x}_{k-1})$ and the observation $P(\boldsymbol{z}_k \mid \boldsymbol{x}_k)$. These need to be presented in a form that allows instantiation of values for $\boldsymbol{z}_k$, $\boldsymbol{x}_k$, and $\boldsymbol{x}_{k-1}$. For low-dimensional state spaces, interpolation in a lookup table is a viable representation. For high-dimensional state spaces, the preferred method is to provide a representation in terms of a function.

Practically, (35.46) and (35.49) are implemented as follows.

Time Update. A process model is defined in the usual state-space form as $\boldsymbol{x}_k = \boldsymbol{f}(\boldsymbol{x}_{k-1}, \boldsymbol{w}_{k-1}, k)$, where $\boldsymbol{w}_k$ is an independent noise sequence with known probability density $P(\boldsymbol{w}_k)$. The prediction step is now implemented as follows: N_k samples $\boldsymbol{w}_k^i$, $i = 1, \cdots, N_k$ are drawn from the distribution $P(\boldsymbol{w}_k)$. The N_k support values $\boldsymbol{x}_{k-1}^i$ together with the samples $\boldsymbol{w}_k^i$ are passed through the process model as

$$\boldsymbol{x}_k^i = \boldsymbol{f}\left(\boldsymbol{x}_{k-1}^i, \boldsymbol{w}_{k-1}^i, k\right) , \tag{35.50}$$

yielding a new set of support vectors $\boldsymbol{x}_k^i$. The weights for these support vectors w_{k-1}^i are not changed. In effect, the process model is simply used to do N_k simulations of state propagation.

Avidbots Ex1016
Page 911 of 2154

Observation Update. The observation model is also defined in the usual state-space form as $\boldsymbol{z}_k = \boldsymbol{h}(\boldsymbol{x}_k, \boldsymbol{v}_k, k)$, where $\boldsymbol{v}_k$ is an independent noise sequence with known probability density $P(\boldsymbol{v}_k)$. The observation step is now implemented as follows. A measurement $\boldsymbol{z}_k = z_k$ is made. For each support value $\boldsymbol{x}_k^i$, a likelihood is computed as

$$\Lambda\left(\boldsymbol{x}_k^i\right) = P\left(\boldsymbol{z}_k = z_k \mid \boldsymbol{x}_k = \boldsymbol{x}_k^i\right) . \tag{35.51}$$

Practically, this requires that the observation model be in an equational form (such as a Gaussian) which allows computation of the likelihood in the error between the measured value z_k and the observations predicted by each particle $\boldsymbol{h}(\boldsymbol{x}_k^i, k)$. The updated weights after observation are just

$$w_k^i \propto w_{k-1}^i P\left(\boldsymbol{z}_k = z_k \mid \boldsymbol{x}_k^i\right) . \tag{35.52}$$

Resampling

After the weights are updated it is usual to resample the measure $\{\boldsymbol{x}_k^i, w_k^i\}_{i=1}^N$. This focuses the samples in on those areas that have most probability density. The decision to resample is made on the basis of the *effective number*, N_{eff} of particles in the sample, approximately estimated from

$$N_{\text{eff}} = \frac{1}{\sum_i (w_k^i)^2} .$$

The sampling importance resampling (SIR) algorithm resamples at every cycle so that the weights are always equal. One of the key problems with resampling is that the sample set fixates on a few highly likely samples. This problem of fixating on a few highly likely particles during resampling is known as *sample impoverishment*. Generally, it is good to resample when N_{eff} falls to some fraction of the actual samples (say $1/2$).

When to Use Monte Carlo Methods

Monte Carlo (MC) methods are well suited to problems where state-transition models and observation models are highly nonlinear. This is because sample-based methods can represent very general probability densities. In particular multimodal or multiple hypothesis density functions are well handled by Monte Carlo techniques. One caveat to note however is that the models $P(\boldsymbol{x}_k \mid \boldsymbol{x}_{k-1})$ and $P(\boldsymbol{z}_k \mid \boldsymbol{x}_k)$ must be enumerable in all cases and must typically be in a simple parametric form. MC methods also span the gap between parametric and grid-based data fusion methods.

Monte Carlo methods are inappropriate for problems where the state space is of high dimension. In general the number of samples required to faithfully model a given density increases exponentially with state-space dimension. The effects of dimensionality can be limited by marginalizing out states that can be modeled without sampling, a procedure known as Rao–Blackwellization.

35.1.5 Alternatives to Probability

The representation of uncertainty is so important to the problem of information fusion that a number of alternative modeling techniques have been proposed to deal with perceived limitations in probabilistic methods.

There are three main perceived limitations of probabilistic modeling techniques:

1. Complexity: the need to specify a large number of probabilities to be able to apply probabilistic reasoning methods correctly.
2. Inconsistency: the difficulties involved in specifying a consistent set of beliefs in terms of probability and using these to obtain consistent deductions about states of interest.
3. Precision of models: the need to be precise in the specification of probabilities for quantities about which little is known.
4. Uncertainty about uncertainty: the difficulty in assigning probability in the face of uncertainty, or ignorance about the source of information.

There are three main techniques put forward to address these issues: interval calculus, fuzzy logic, and the theory of evidence (Dempster–Shafer methods). We briefly discuss each of these in turn.

Interval Calculus

The representation of uncertainty using an interval to bound true parameter values has a number of potential advantages over probabilistic techniques. In particular, intervals provide a good measure of uncertainty in situations where there is a lack of probabilistic information, but in which sensor and parameter error is known to be bounded. In interval techniques, the uncertainty in a parameter x is simply described by a statement that the true value of the state x is known to be bounded from below by a, and from above by b; $x \in [a, b]$. It is important that no other additional probabilistic structure is implied; in particular the statement $x \in [a, b]$ does not necessarily imply that x is equally probable (uniformly distributed) over the interval $[a, b]$.

There are a number of simple and basic rules for the manipulation of interval errors. These are described in detail in the book by *Moore* [35.13] (whose analysis was originally aimed at understanding limited-precision computer arithmetic). Briefly, with $a, b, c, d \in \mathbb{R}$, addi-

Avidbots Ex1016
Page 912 of 2154

tion, subtraction, multiplication, and division are defined by the following algebraic relations

$$[a,b]+[c,d]=[a+c,b+d]\,,$$
$$[a,b]-[c,d]=[a-d,b-c]\,, \tag{35.53}$$
$$[a,b]\times[c,d]=[\min(ac,ad,bc,bd),\ \max(ac,ad,bc,bd)]\,, \tag{35.54}$$
$$\frac{[a,b]}{[c,d]}=[a,b]\times\left[\frac{1}{d},\frac{1}{c}\right]\,,\quad 0\notin[c,d]\,. \tag{35.55}$$

It can be shown that interval addition and multiplication are both associative and commutative. Interval arithmetic admits an obvious metric distance measure

$$d([a,b],[c,d])=\max(|a-c|,|b-d|)\,. \tag{35.56}$$

Matrix arithmetic using intervals is also possible, but substantially more complex, particularly when matrix inversion is required.

Interval calculus methods are sometimes used for detection. However, they are not generally used in data fusion problems since:

1. It is difficult to get results that converge to anything of value (it is too pessimistic), and
2. It is hard to encode dependencies between variables which are at the core of many data fusion problems.

Fuzzy Logic

Fuzzy logic has found widespread popularity as a method for representing uncertainty particularly in applications such as supervisory control and high-level data fusion tasks. It is often claimed that fuzzy logic provides an ideal tool for inexact reasoning, particularly in rule-based systems. Certainly, fuzzy logic has had some notable success in practical application.

A great deal has been written about fuzzy sets and fuzzy logic (e.g. [35.14] and the discussion in [35.15, Chap. 11]). Here we briefly describe the main definitions and operations without any attempt to consider the more advanced features of fuzzy-logic methods.

Consider a universal set consisting of the elements x; $\mathcal{X}=\{x\}$. Consider a proper subset $\mathcal{A}\subseteq\mathcal{X}$ such that

$$\mathcal{A}=\{x\mid x \text{ has some specific property}\}\,.$$

In conventional logic systems, we can define a membership function $\mu_A(x)$ (also called the *characteristic function* which reports if a specific element $x\in\mathcal{X}$ is a member of this set

$$\mathcal{A}\rightleftharpoons\mu_A(x)=\begin{cases}1 & \text{if } x\in A\\ 0 & \text{if } x\notin A\end{cases}\,.$$

For example $\mathcal{X}$ may be the set of all aircraft. The set $\mathcal{A}$ may be the set of all supersonic aircraft. In the fuzzy logic literature, this is known as a *crisp* set. In contrast, a fuzzy set is one in which there is a *degree of membership*, ranging between 0 and 1. A fuzzy membership function $\mu_A(x)$ then defines the degree of membership of an element $x\in\mathcal{X}$ to the set $\mathcal{A}$. For example, if $\mathcal{X}$ is again the set of all aircraft, $\mathcal{A}$ may be the set of all *fast* aircraft. Then the fuzzy membership function $\mu_A(x)$ assigns a value between 0 and 1 indicating the degree of membership of every aircraft x to this set. Formally

$$\mathcal{A}\rightleftharpoons\mu_A\mapsto[0,1]\,.$$

Composition rules for fuzzy sets follow the composition processes for normal crisp sets, for example,

$$\mathcal{A}\cap\mathcal{B}\rightleftharpoons\mu_{A\cap B}(x)=\min[\mu_A(x),\mu_B(x)]\,,$$
$$\mathcal{A}\cup\mathcal{B}\rightleftharpoons\mu_{A\cup B}(x)=\max[\mu_A(x),\mu_B(x)]\,.$$

The normal properties associated with binary logic now hold: commutativity, associativity, idempotence, distributivity, De Morgan's law, and absorption. The only exception is that the law of the excluded middle is no longer true

$$\mathcal{A}\cup\overline{\mathcal{A}}\neq\mathcal{X}\,,\quad \mathcal{A}\cap\overline{\mathcal{A}}\neq\emptyset\,.$$

Together these definitions and laws provide a systematic means of reasoning about inexact values.

The relationship between fuzzy set theory and probability is still hotly debated.

Evidential Reasoning

Evidential reasoning (often called the Dempster–Shafer theory of evidence after the originators of these ideas) has seen intermittent success particularly in automated reasoning applications. Evidential reasoning is qualitatively different from either probabilistic methods or fuzzy set theory in the following sense. Consider a universal set $\mathcal{X}$. In probability theory or fuzzy set theory, a belief mass may be placed on any element $x_i\in\mathcal{X}$ and indeed on any subset $\mathcal{A}\subseteq\mathcal{X}$. In evidential reasoning, belief mass can not only be placed on elements and sets, but also sets of sets. Specifically, while the domain of probabilistic methods is all possible subsets $\mathcal{X}$, the domain of evidential reasoning is the power set $2^{\mathcal{X}}$.

As an example, consider the mutually exclusive set $\mathcal{X}=\{\text{occupied},\text{empty}\}$. In probability theory we

Avidbots Ex1016
Page 913 of 2154

might assign a probability to each possible event, for example, $P(\text{occupied}) = 0.3$, and thus $P(\text{empty}) = 0.7$. In evidential reasoning, we construct the set of all subsets

$$2^{\mathcal{X}} = \{\{\text{occupied}, \text{empty}\}, \{\text{occupied}\}, \{\text{empty}\}, \emptyset\} ,$$

and belief mass is assigned to all elements of this set as

$$\begin{aligned} m(\{\text{occupied}, \text{empty}\}) &= 0.5 , \\ m(\{\text{occupied}\}) &= 0.3 , \\ m(\{\text{empty}\}) &= 0.2 , \\ m(\emptyset) &= 0.0 , \end{aligned}$$

(the null set $\emptyset$ is assigned a belief mass of zero for normalization purposes). The interpretation of this is that there is a 30% chance of occupied, a 20% chance of empty, and a 50% chance of either occupied or empty. In effect, the measure placed on the set containing both occupied and empty is a measure of ignorance or inability to distinguish between the two alternatives. See [35.16] for a more-detailed example of applying the evidential method to certainty-grid navigation.

Evidential reasoning thus provides a method of capturing ignorance or an inability to distinguish between alternatives. In probability theory, this would be dealt with in a very different manner by assigning an equal or uniform probability to each alternative. Yet, stating that there is a 50% chance of occupancy is clearly *not* the same as saying that it is unknown whether it will occupied or not. The use of the power set as the *frame of discernment* allows a far richer representation of beliefs. However, this comes at the cost of a substantial increase in complexity. If there are n elements in the original set $\mathcal{X}$, then there will be 2^n possible subsets on which a belief mass will be assigned. For large n, this is clearly intractable. Further, when the set is continuous, the set of all subsets is not even measurable.

Evidential reasoning methods provide a means of assigning, and combining belief masses on sets. Methods also exist for obtaining related measures called *support* and *plausibility* which, in effect, provide upper and lower probability bounds in agreement with Dempster's original formulation of this method.

Evidential reasoning can play an important role in discrete data fusion systems, particularly in areas such as attribute fusion and situation assessment, where information may be unknown or ambiguous. Its use in lower-level data fusion problems is challenging as the assignment of belief mass to the power set scales exponentially with state cardinality.

35.2 Multisensor Fusion Architectures

The multisensor fusion methods described in the previous section provide the algorithmic means by which sensor data and their associated uncertainty models can be used to construct either implicit or explicit models of the environment. However, a multisensor fusion system must include many other functional components to manage and control the fusion process. The organization of these is termed a *multisensor fusion architecture*.

35.2.1 Architectural Taxonomy

Multisensor systems architectures can be organized in various ways. The military community has developed a layout of functional architectures based on the joint directors of the laboratories (JDL) model for multisensor systems. This approach views multisensor fusion in terms of signal, feature, threat and situation analysis levels (so-called JDL levels). The assessment of such systems is specified in terms of tracking performance, survivability, efficiency and bandwidth. Such measures are not generally appropriate in robotics applications and so the JDL model is not discuss this further here (see [35.17, 18] for details). Other classification schemes distinguish between low- and high-level fusion [35.19], or centralized versus decentralized processing or data versus variable [35.20].

A general architectural framework for multisensor robotic systems has been developed and described in detail by *Makarenko* [35.21], and we will base our discussion on this approach. A system architecture is defined as follows.

Meta-Architecture

A set of high-level considerations that strongly characterize the system structure. The selection and organization of the system elements may be guided by aesthetics, efficiency, or other design criteria and goals (for example, system and component comprehensibility, modularity, scalability, portability, interoperability, (de)centralization, robustness, fault tolerance).

Algorithmic Architecture

A specific set of information fusion and decision-making methods. These methods address data heterogeneity, registration, calibration, consistency, informa-

tion content, independence, time interval and scale, and relationships between models and uncertainty.

Conceptual Architecture

The granularity and functional roles of components (specifically, mappings from algorithmic elements to functional structures).

Logical Architecture

Detailed canonical component types (i. e., object-oriented specifications) and interfaces to formalize intercomponent services. Components may be ad hoc or regimented, and other concerns include granularity, modularity, reuse, verification, data structures, semantics, etc. Communication issues include hierarchical versus heterarchical organization, shared memory versus message passing, information-based characterizations of subcomponent interactions, pull/push mechanisms, subscribe–publish mechanisms, etc. Control involves both the control of actuation systems within the multisensor fusion system, as well as control of information requests and dissemination within the system, and any external control decisions and commands.

Execution Architecture

Defines mapping of components to execution elements. This includes internal or external methods of ensuring correctness of the code (i. e., that the environment and sensor models have been correctly transformed from mathematical or other formal descriptions into computer implementations), and also validation of the models (i. e., ensure that the formal descriptions match physical reality to the required extent).

In any closed-loop control system, sensors are used to provide the feedback information describing the current status of the system and its uncertainties. Building a sensor system for a given application is a system engineering process that includes the analysis of system requirements, a model of the environment, the determination of system behavior under different conditions, and the selection of suitable sensors [35.22]. The next step in building the sensor system is to assemble the hardware components and develop the necessary software modules for data fusion and interpretation. Finally, the system is tested, and the performance is analyzed. Once the system is built, it is necessary to monitor the different components of the system for the purpose of testing, debugging, and analysis. The system also requires quantitative measures in terms of time complexity, space complexity, robustness, and efficiency.

In addition, designing and implementing real-time systems are becoming increasingly complex owing to many added features such as graphical user interfaces (GUIs), visualization capabilities, and the use of many sensors of different types. Therefore, many software engineering issues such as reusability and the use of commercial off-the-shelf (COTS) components [35.23], real-time issues [35.24–26], sensor selection [35.27], reliability [35.28–30], and embedded testing [35.31] are now receiving more attention from system developers.

Each sensor type has different characteristics and functional descriptions. Consequently, some approaches aim to develop general methods of modeling sensor systems in a manner that is independent of the physical sensors used. In turn, this enables the performance and robustness of multisensor systems to be studied in a general way. There have been many attempts to provide *the* general model, along with its mathematical basis and description. Some of these modeling techniques concern error analysis and fault tolerance of multisensor systems [35.32–37]. Other techniques are model based, and require a priori knowledge of the sensed object and its environment [35.38–40]. These help fit data to a model, but do not always provide the means to compare alternatives. Task-directed sensing is another approach to devising sensing strategies [35.41–43]. General sensor modeling work has had a considerable influence on the evolution of multisensor fusion architectures.

Another approach to modeling sensor systems is to define sensori-computational systems associated with each sensor to allow design, comparison, transformation, and reduction of any sensory system [35.44]. In this approach, the concept of an information invariant is used to define a measure of information complexity. This provides a computational theory allowing analysis, comparison, and reduction of sensor systems.

In general terms, multisensor fusion architectures may be classified according to the choice along four independent design dimensions:

1. Centralized–decentralized,
2. Local–global interaction of components,
3. Modular–monolithic, and
4. Heterarchical–hierarchical.

The most prevalent combinations are:

- Centralized, global interaction, and hierarchical,
- Decentralized, global interaction, and heterarchical,
- Decentralized, local interaction, and hierarchical,
- Decentralized, local interaction, and heterarchical.

In some cases explicit modularity is also desirable. Most existing multisensor architectures fit reasonably well into one of these categories. These categories make no general commitment to the algorithmic architecture.

Avidbots Ex1016
Page 915 of 2154

If the algorithmic architecture is the predominant feature of a system, it will be characterized as part of multisensor fusion theory in Sect. 35.1; otherwise, it merely differentiates methods within one of the four meta-architectures.

35.2.2 Centralized, Local Interaction, and Hierarchical

Centralized, local interaction and hierarchical architectures encompass a number of system philosophies. Least representationally demanding is the *subsumption architecture* initially proposed by *Braitenberg* [35.45] and popularized by *Brooks* [35.46]. The subsumption multisensor architecture defines behaviors as the basic components, and employs a layered set of behaviors to embody one program (monolithic). Any behavior may utilize the output of other behaviors, and may also inhibit other behaviors. The hierarchy is defined by the layers, although this is not always clear-cut. The major design philosophy is to develop behaviors directly from perception–action loops without recourse to brittle, environment representations. This leads to robustness in operation, but a lack of composite behavior predictability.

A more sophisticated (representationally) behavior-based system is the *distributed field robot architecture* (DFRA) [35.47]. This is a generalization of the sensor fusion effects (SFX) architecture [35.48]. This approach exploits modularity, and aims to achieve both behavior-based and deliberative action, reconfigurability and interoperability through the use of Java, Jini, and XML, fault tolerance, adaptability, longevity, consistent interfaces, and dynamic components. The algorithmic architecture is based on fuzzy logic controllers. Experiments have been demonstrated on outdoor mobile robot navigation.

Other similar architectures of this type include perception action networks *Lee* and *Ro* [35.49, 50], while *Draper* et al. [35.51] focuses on types of information needed to perform tasks (higher-level integration); see also [35.52].

Another approach to this type of sensor fusion is to use artificial neural networks. The advantage is that the user, at least in principle, does not need to understand how sensor modalities relate, nor model the uncertainties, nor in fact determine the structure of the system more than to specify the number of layers in the network and the number of nodes per layer. The neural network is presented with a set of training examples, and must determine through the weights on the neuron connections the optimal mapping from inputs to desired outputs (classifications, control signals, etc.) [35.53, 54].

Various other methods exist; for example, *Hager* and *Mintz* [35.42, 43] defines a task-oriented approach to sensor fusion based on Bayesian decision theory and develops an object-oriented programming framework. *Joshi* and *Sanderson* [35.55] describe a

> *methodology for addressing model selection and multisensor fusion issues using representation size (description length) to choose (1) model class and number of parameters, (2) model parameter resolution (3) subset of observed features to model, and (4) correspondence to map features to models.*

Their approach is broader than an architecture and uses a minimization criterion to synthesize a multisensor fusion system to solve specific two-dimensional (2-D) and three-dimensional (3-D) object recognition problems.

35.2.3 Decentralized, Global Interaction, and Heterarchical

The major example of the decentralized, global interaction meta-architecture is the blackboard system. There have been many examples of blackboard systems developed for data fusion applications. For example, the SEPIA system of *Berge-Cherfaoui* and *Vachon* [35.56] uses logical sensors (see below) in the form of modular agents that post results to a blackboard. The overall architectural goals for blackboards include efficient collaboration and dynamic configuration. Experiments are reported on an indoor robot moving from room to room.

The MESSIE (multi expert system for scene interpretation and evaluation) system [35.57] is a scene interpretation system based on multisensor fusion; it has been applied to the interpretation of remotely sensed images. A typology of the multisensor fusion concepts is presented, and the consequences of modeling problems for objects, scene, and strategy are derived. The proposed multispecialist architecture generalized the ideas of their previous work by taking into account the knowledge of sensors, the multiple viewing notion (shot), and the uncertainty and imprecision of models and data modeled with possibility theory. In particular, generic models of objects are represented by concepts independent of sensors (geometry, materials, and spatial context). Three kinds of specialists are present in the architecture: generic specialists (scene and conflict), semantic object specialists, and low-level specialists. A blackboard structure with a centralized control is used. The interpreted scene is implemented as a matrix of pointers enabling conflicts to be detected very easily. Under the control of the scene specialist, the conflict

Avidbots Ex1016
Page 916 of 2154

specialist resolves conflicts using the spatial context knowledge of objects. Finally, an interpretation system with SAR (synthetic aperture radar)/SPOT sensors is described, and an example of a session concerned with bridge, urban area, and road detection is shown.

35.2.4 Decentralized, Local Interaction, and Hierarchical

One of the earliest proposals for this type of architecture is the real-time control system (RCS) [35.58]. RCS is presented as a cognitive architecture for intelligent control, but essentially uses multisensor fusion to achieve complex control. RCS focuses on task decomposition as the fundamental organizing principle. It defines a set of nodes, each comprised of a sensor processor, a world model, and a behavior generation component. Nodes communicate with other nodes, generally in a hierarchical manner, although across-layer connections are allowed. The system supports a wide variety of algorithmic architectures, from reactive behavior to semantic networks. Moreover, it maintains signals, images, and maps, and allows tight coupling between iconic and symbolic representations. The architecture does not generally allow dynamic reconfiguration, but maintains the static module connectivity structure of the specification. RCS has been demonstrated in unmanned ground vehicles [35.59]. Other object-oriented approaches have been reported [35.34, 60].

An early architectural approach which advocated strong programming semantics for multisensor systems is the logical sensor system (LSS). This approach exploits functional (or applicative) language theory to achieve that.

The most developed version of LSS is instrumented LSS (ILLS) [35.22]. The ILLS approach is based on the LSS introduced by *Shilcrat* and *Henderson* [35.61]. The LSS methodology is designed to specify any sensor in a way that hides its physical nature. The main goal behind LSS was to develop a coherent and efficient presentation of the information provided by many sensors of different types. This representation provides a means for recovery from sensor failure, and also facilitates reconfiguration of the sensor system when adding or replacing sensors [35.62].

ILSS is defined as an extension to LSS, and is comprised of the following components (Fig. 35.4):

1. *ILS* name: uniquely identifies a module
2. *Characteristic output vector (COV)*: strongly typed output structure, with one output vector and zero or more input vectors
3. *Commands*: input commands to the module, and output commands to the other modules
4. *Select function*: a selector that detects the failure of an alternate and switches to another alternate if possible
5. *Alternate subnets*: alternative ways of producing the COV_{out}; it is these implementations of one or more algorithms that carry the main functions of the module
6. *Control command interpreter (CCI)*: interpreter of the commands to the module
7. *Embedded tests*: self-testing routines that increase robustness and facilitate debugging
8. *Monitors*: modules that check the validity of the resulting COVs
9. *Taps*: hooks on the output lines to view different COV values.

These components identify the system behavior and provide mechanisms for online monitoring and debugging. In addition, they provide handles for measuring the runtime performance of the system. Monitors are validity check stations that filter the output and alert the user to any undesired results. Each monitor is equipped with a set of rules (or constraints) that governs the behavior of the COV under different conditions.

Embedded testing is used for online checking and debugging purposes. *Weller* et al. proposed a sensor-processing model with the ability to detect measurement errors and to recover from these errors [35.31]. This method is based on providing each system module with verification tests to verify certain characteristics in the measured data, and to verify the internal and output data resulting from the sensor-module algorithm. The recovery strategy is based on rules that are local to the

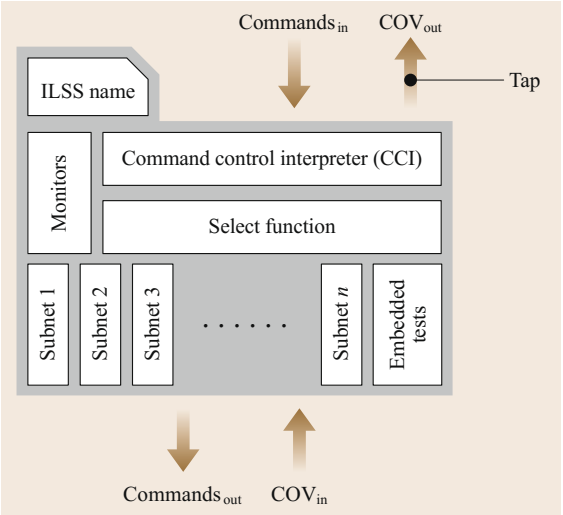


Fig. 35.4 Instrumented logical sensor module

Avidbots Ex1016
Page 917 of 2154

Table 35.1 Canonical components and the roles they play. Multiple ✓ in the same row indicate that some interrole relationships are internalized within a component. *Frame* does not participate in information fusion or decision making but is required for localization and other platform-specific tasks (after [35.21])

Component	Belief			Plan			Action	
Type	Source	Fuse/dist	Sink	Source	Fuse/dist	Sink	Source	Sink
Sensor	✓							
Node		✓			✓			
Actuator								✓
Planner			✓	✓		✓	✓	
UI	✓		✓			✓	✓	
Frame								

different sensor modules. ILSS uses a similar approach called *local embedded testing*, in which each module is equipped with a set of tests based on the semantic definition of that module. These tests generate input data to check different aspects of the module, then examine the output of the module using a set of constraints and rules defined by the semantics. These tests can also take input from other modules to check the operation of a group of modules. Examples are given of a wall-posed estimation system comprised of a Labmate platform with a camera and sonars. Many extensions have been proposed for LSS [35.63, 64].

35.2.5 Decentralized, Local Interaction, and Heterarchical

The best example of this meta-architecture is the active sensor network (ASN) framework for distributed data fusion developed by *Makarenko* et al. [35.21, 65]. The distinguishing features of the various architectures are now described.

Meta-architecture

The distinguishing features of ASN are its commitment to decentralization, modularity, and strictly local interactions (this may be physical or by type). Thus, these are communicating processes. Decentralized means that no component is central to operation of the system, and the communication is peer to peer. Also, there are no central facilities or services (e.g., for communication, name and service lookup or timing). These features lead to a system that is scalable, fault tolerant, and reconfigurable.

Local interactions mean that the number of communication links does not change with the network size. Moreover, the number of messages should also remain constant. This makes the system scalable as well as reconfigurable.

Modularity leads to interoperability derived from interface protocols, reconfigurability, and fault tolerance: failure may be confined to individual modules.

Algorithmic Architecture

There are three main algorithmic components: belief fusion, utility fusion, and policy selection. Belief fusion is achieved by communicating all beliefs to neighboring platforms. A belief is defined as a probability distribution of the world state space.

Utility fusion is handled by separating the individual platform's partial utility into the team utility of belief quality and local utilities of action and communication. The downside is that the potential coupling between individual actions and messages is ignored because the utilities of action and communication remain local.

The communication and action policies are chosen by maximizing expected values. The selected approach is to achieve point maximization for one particular state and follows the work of *Durrant-Whyte* et al. [35.11, 66].

Conceptual Architecture

The data types of the system include:

1. *Beliefs*: Current world beliefs
2. *Plans*: Future planned world beliefs
3. *Actions*: Future planned actions.

The definition of component roles leads to a natural partition of the system.

The information fusion task is achieved through the definition of four component roles for each data type: *source*, *sink*, *fuser*, and *distributor*. (Note that the data type action does not have fuser or distributor component roles.)

Connections between distributors form the backbone of the ASN framework, and the information exchanged is in the form of their local beliefs. Similar considerations are used to determine component roles for the decision-making and system configuration tasks.

Logical Architecture

A detailed architecture specification is determined from the conceptual architecture. It is comprised of

Avidbots Ex1016
Page 918 of 2154

six canonical component types as described in Table 35.1 [35.21].

Makarenko then describes how to combine the components and interfaces to realize the use cases of the problem domain in ASN.

Execution Architecture

The execution architecture traces the mapping of logical components to runtime elements such as processes and shared libraries. The deployment view shows the mapping of physical components onto the nodes of the physical system. The source code view explains how the software implementing the system is organized. At the architectural level, three items are addressed: execution, deployment, and source code organization.

The experimental implementation of the ASN framework has proven to be flexible enough to accommodate a variety of system topologies, platform, and sensor hardware, and environment representations. Several examples are given with a variety of sensors, processors, and hardware platforms.

35.3 Applications

Multisensor fusion systems have been applied to a wide variety of problems in robotics (see the references for this chapter and VIDEO 132, VIDEO 638 and VIDEO 639), but the two most general areas are *dynamic system control* and *environment modeling*. Although there is some overlap between these, they may generally be characterized as:

- *Dynamic system control*: The problem is to use appropriate models and sensors to control the state of a dynamic system (e.g., industrial robot, mobile robot, autonomous vehicle, surgical robot). Usually such systems involve real-time feedback control loops for steering, acceleration, and behavior selection. In addition to state estimation, uncertainty models are required. Sensors may include force/torque sensors, gyros, global positioning system (GPS), position encoders, cameras, range finders, etc..
- *Environment modeling*: The problem is to use appropriate sensors to construct a model of some aspect of the physical environment. This may be a particular object, e.g., a cup, a physical part, a face, or a larger part of the surroundings, e.g., the interior of a building, part of a city or an extended remote or underground area. Typical sensors include cameras, radar, 3-D range finders, infrared (IR), tactile sensors and touch probes (CMMs), etc. The result is usually expressed as geometry (points, lines, surfaces), features (holes, sinks, corners, etc.), or physical properties. Part of the problem includes the determination of optimal sensor placement.

35.3.1 Dynamic System Control

The EMS-Vision system [35.67] is an outstanding exemplar of this application domain. The goal is to develop a robust and reliable perceptual system for autonomous vehicles. The development goals, as stated by the EMS-Vision team, are:

- COTS components
- Wide variety of objects modeled and incorporated into behaviors
- Inertial sensors for ego-state estimation
- Peripheral/foveal/saccadic vision
- Knowledge and goal driven behavior
- State tracking for objects
- 25 Hz real-time update rate.

The approach has been in development since the 1980s. Figure 35.5 shows the first vehicle to drive fully autonomously on the German autobahn for 20 km and at speeds up to 96 km/h.

Information from inertial and vision sensors is combined to produce a road scene tree (Fig. 35.6). A four-



Fig. 35.5 First fully autonomous vehicle on German autobahn

Avidbots Ex1016
Page 919 of 2154

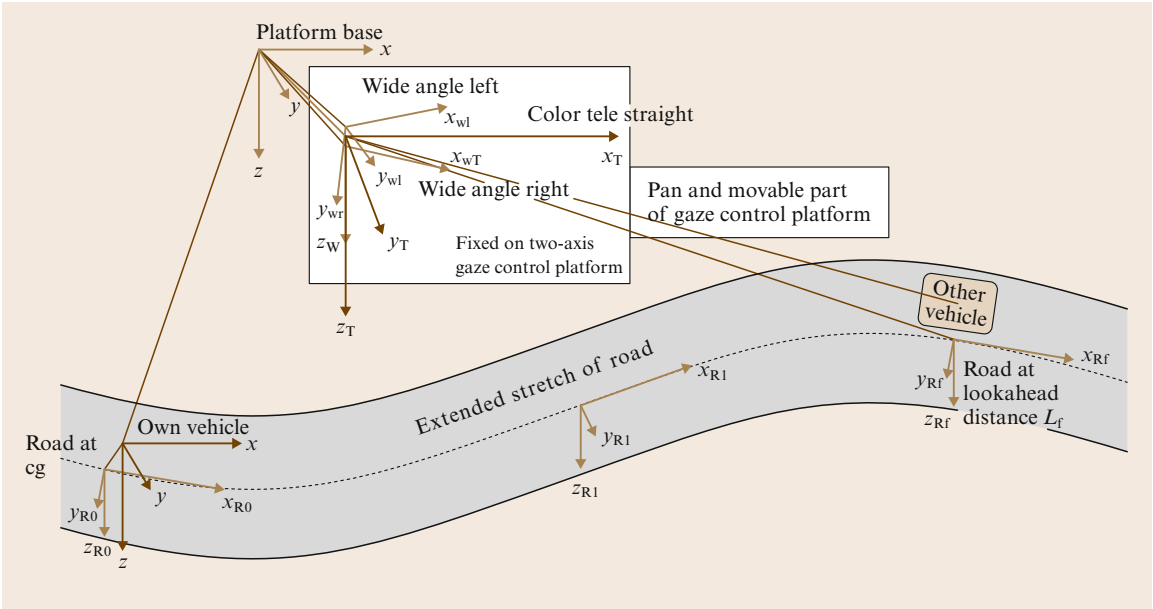


Fig. 35.6 EMS-Vision road scene tree

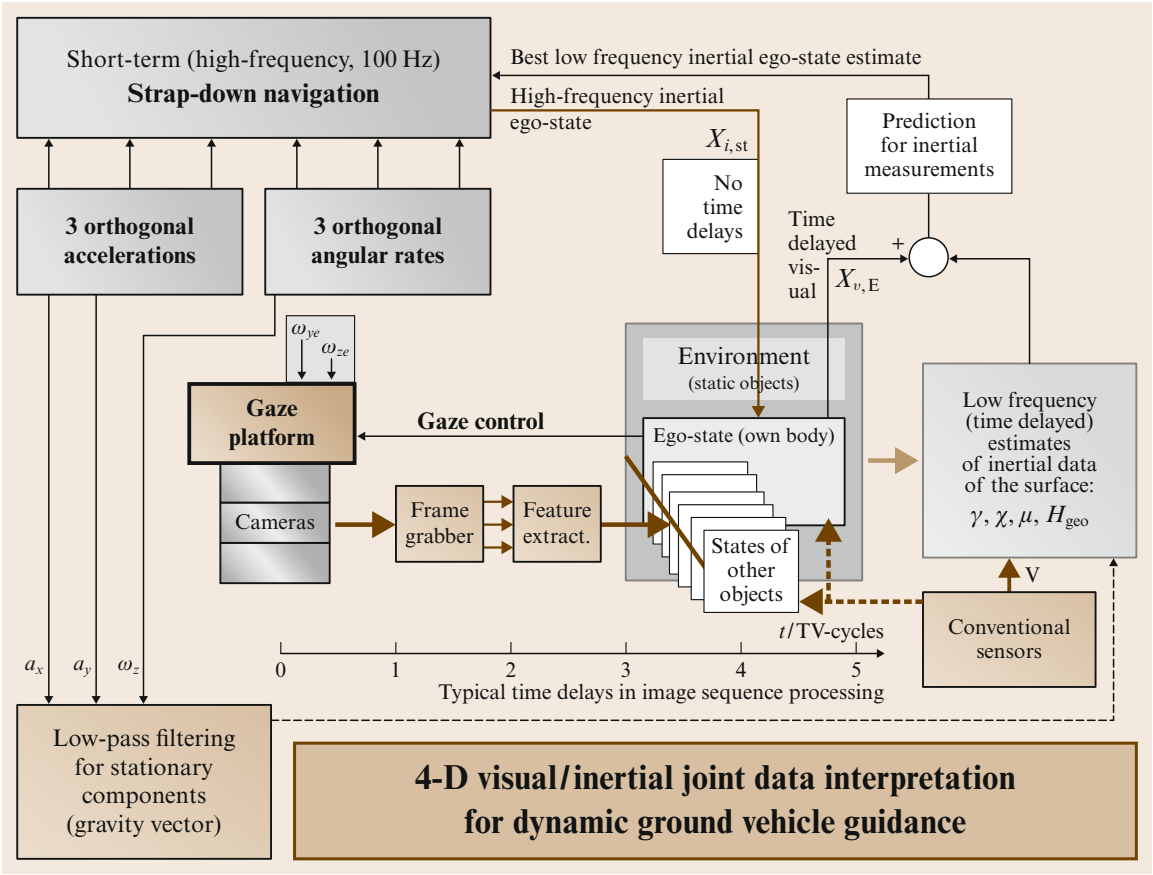


Fig. 35.7 EMS-Vision guidance system

Avidbots Ex1016
Page 920 of 2154

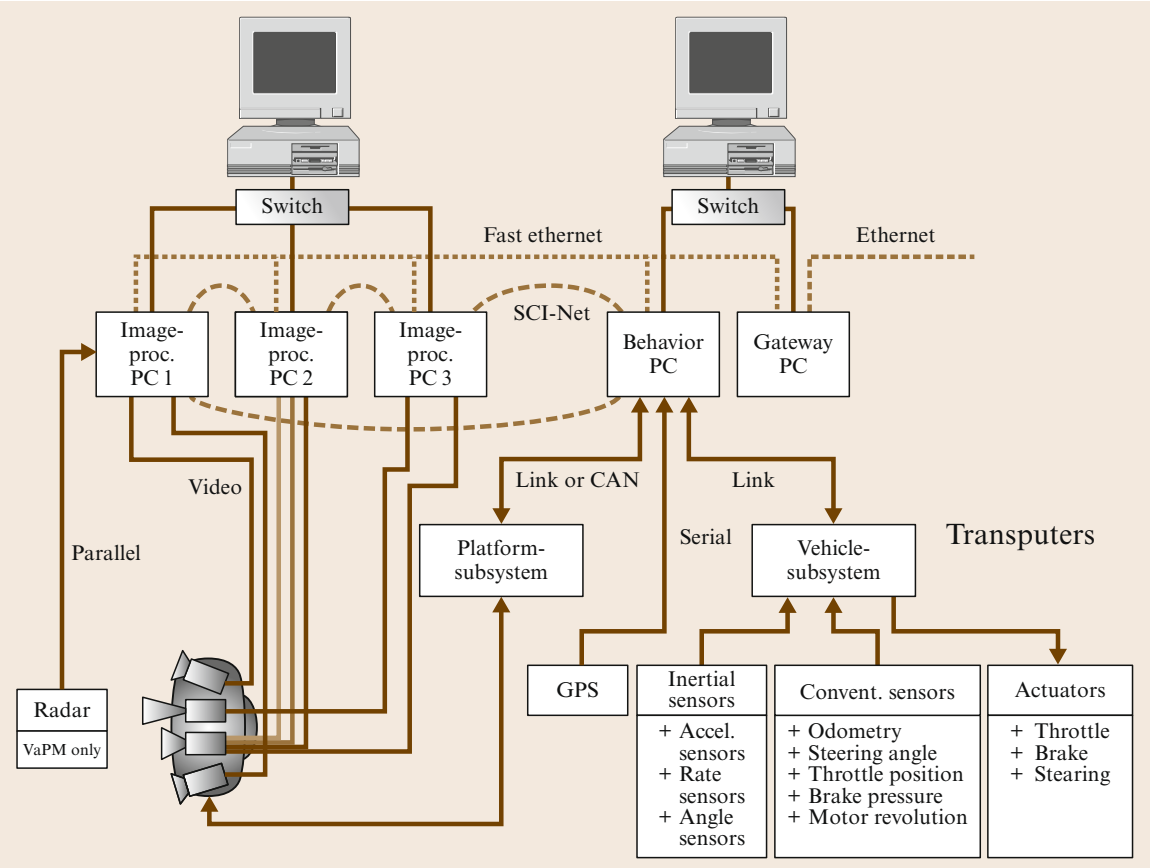


Fig. 35.8 EMS-Vision hardware layout (CAN: controller area network)

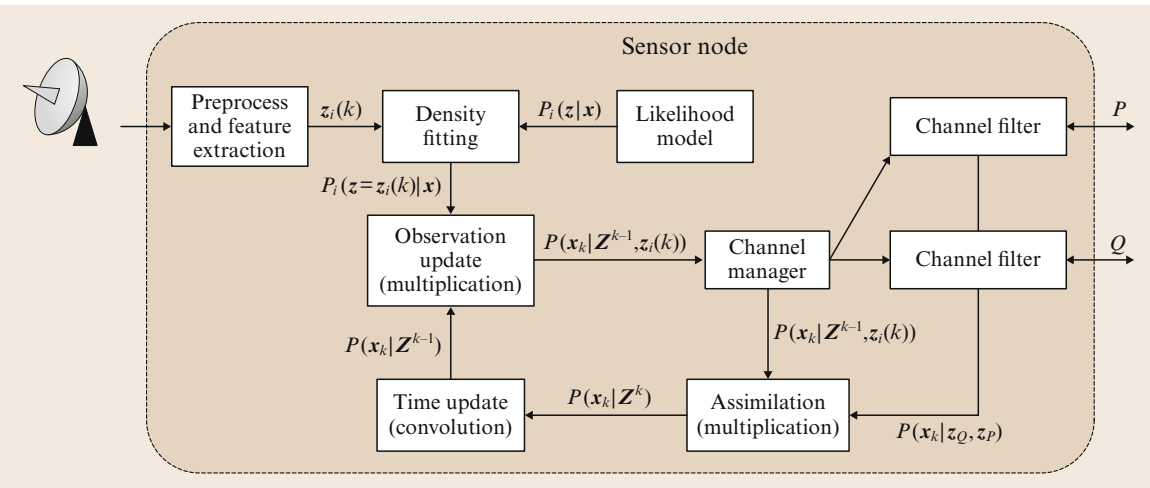


Fig. 35.9 Mathematical structure of a decentralized data fusion node

Avidbots Ex1016
Page 921 of 2154

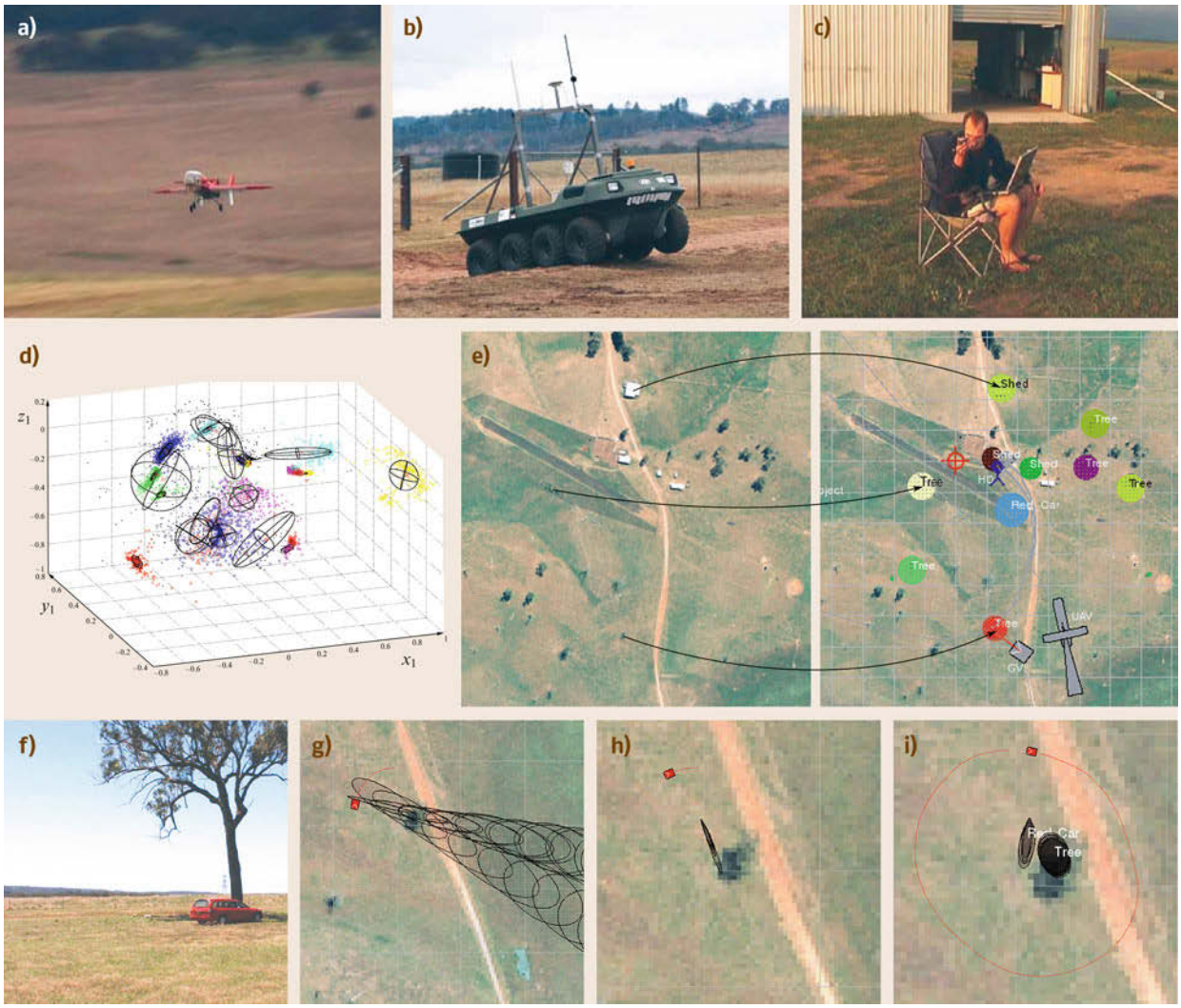


Fig.35.10a–i A synopsis of the ANSER II autonomous network and its operation. **(a–c)** Main system components: **(a)** air vehicle, **(b)** ground vehicle, **(c)** human operative. **(d–e)** The perception process: **(d)** top three dimensions of features discovered from ground-based visual sensor data along with the derived mixture model describing these feature properties, **(e)** sector of the overall picture obtained from fusing air vehicle (UAV), ground vehicle (GV), and human operator (HO) information. Each set of ellipses corresponds to a particular feature and the labels represent the identity state with highest probability. **(f–i)** Sequential fusion process for two close landmarks: **(f)** a tree and a red car, **(g)** bearing-only visual observations of these landmarks are successively fused, **(h)** to determine location and identity **(i)**. Note the Gaussian mixture model for the bearing measurement likelihood

dimensional (4-D) generic object representation is built which includes background knowledge of the object (e.g., roads), its behavioral capabilities, object state and variances, and shape and aspect parameters. Figure 35.7 shows the 4-D inertial/vision multisensor guidance system, while Fig. 35.8 shows the hardware aspects.

In summary, the EMS-Vision system is an interesting and powerful demonstration of multisensor fusion for dynamic system control.

35.3.2 ANSER II: Decentralized Data Fusion

Decentralized data fusion (DDF) methods were initially motivated by the insight that the *information* or canonical form of the conventional Kalman filter data fusion algorithm could be implemented by simply adding information contributions from observations as shown in (35.41). As these (vector and matrix) additions are commutative, the update or data fusion process can

Avidbots Ex1016
Page 922 of 2154

be optimally distributed amongst a network of sensors [35.11, 12, 68]. The aim of the ANSER II project was to generalize the DDF method to deal with non-Gaussian probabilities for observations and states, and to incorporate information from a diversity of sources including uninhabited air and ground vehicles, terrain databases, and human operatives.

The mathematical structure of a DDF sensor node is shown in Fig. 35.9. The sensor is modeled directly in the form of a likelihood function. Once instantiated with an observation, the likelihood function is input to a local fusion loop which implements a local form of the Bayesian time and observation update of (35.7) and (35.8). Network nodes accumulate probabilistic information from observation or communication and exchange mutual information (information gain) with other nodes in the network [35.21]. This mutual information is transmitted to and assimilated by other nodes in the network in an ad hoc manner. The result is that all nodes in the network obtain a single integrated posterior-probability-based all-node observations.

The ANSER II system consists of a pair of autonomous air vehicles equipped with infrared and visual sensors, a pair of unmanned ground vehicles equipped with visual and radar sensors, and additional information provided by geometric and hyperspectral databases, along with information input by human operatives [35.69]. The likelihood functions for single-sensor features are obtained through a semisupervised machine learning method [35.70]. The resulting probabilities are modeled in the form of a mixture of Gaussians. Each platform then maintains a bank of decentralized, non-Gaussian Bayesian filters for the observed features, and transmits this information to all other platforms. The net result is that each platform maintains a complete map of all features observed by all nodes in the network. Multiple observations of the same feature, possibly by different platforms, result in an increasingly accurate estimate of the feature location for all nodes. A corresponding discrete probability measure is used for Fig. 35.10 shows a synopsis of the operation of the ANSER II system.

The ANSER II system demonstrates a number of general principles in Bayesian data fusion methods, specifically the need to model sensors appropriately through the likelihood function, and the possibility of building very different data fusion architectures from the essential Bayesian form.

35.3.3 Recent Developments

There have been recent developments in multisensor fusion methods: see [35.71–73] for more theoretical work, vision and biomedical applications [35.74–77], tracking and terrain classification [35.78–81], and robotics and automation [35.82–87].

35.4 Conclusions

Multisensor data fusion has progressed greatly in the last few decades; further advances in the field will be documented in the robotics and multisensor fusion and integration conference and journal literature. Robust applications are being fielded based on the body of theory and experimental knowledge produced by the research community. Current directions of interest include:

1. Large-scale, ubiquitous sensor systems,
2. Bio-based or biomimetic systems,
3. Medical in situ applications,
4. Wireless sensor networks.

Representative large-scale examples include intelligent vehicle and road systems, as well as instrumented contexts such as cities. Biological principles may provide fundamentally distinct approaches to the exploitation of dense, redundant, correlated, noisy sensors, especially when considered as part of a Gibbsian framework for behavioral response to environmental stimuli. Another issue here is the development of a theoretical understanding of sensor system development, adaptivity, and learning with respect to the particular context in which the system is deployed.

Further pushing the envelope of both technology and theory will permit the introduction of micro- and nanosensors into the human body and allow the monitoring and locally adaptive treatment of various illnesses. Finally, a more-complete theoretical framework that encompasses system models for wireless sensor networks is still required. This should include models of the physical phenomena being monitored, as well as operational and network issues. Finally, numerical analysis of the algorithmic properties of data-driven systems with sensor data error sources must be unified with the analysis of truncation, roundoff, and other errors.

A firm foundation exists upon which to build these new theories, systems, and applications. It will be a vibrant area of research for years to come.

Avidbots Ex1016
Page 923 of 2154

Video-References

VIDEO 132 AnnieWay
available from http://handbookofrobotics.org/view-chapter/35/videodetails/132
VIDEO 638 Application of visual odometry for sewer inspection robots
available from http://handbookofrobotics.org/view-chapter/35/videodetails/638
VIDEO 639 Multisensor remote surface inspection
available from http://handbookofrobotics.org/view-chapter/35/videodetails/639

References

35.1 S. Thrun, W. Burgard, D. Fox: *Probabilistic Robotics* (MIT Press, Cambridge 2005)
35.2 J.O. Berger: *Statistical Decision Theory and Bayesian Analysis* (Springer, Berlin, Heidelberg 1985)
35.3 A. Elfes: Sonar-based real-world mapping and navigation, IEEE Trans. Robotics Autom. **3**(3), 249–265 (1987)
35.4 L. Matthies, A. Elfes: Integration of sonar and stereo range data using a grid-based representation, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (1988) pp. 727–733
35.5 L.D. Stone, C.A. Barlow, T.L. Corwin: *Bayesian Multiple Target Tracking* (Artech House, Norwood 1999)
35.6 Y. Bar-Shalom: *Multi-Target Multi-Sensor Tracking* (Artec House, Norwood 1990)
35.7 Y. Bar-Shalom, T.E. Fortmann: *Tracking and Data Association* (Academic, New York 1988)
35.8 P.S. Maybeck: *Stochastic Models, Estimaton and Control* (Academic, New York 1979)
35.9 W. Sorensen: Special issue on the applications of the Kalman filter, IEEE Trans. Autom. Control **28**(3), 254–255 (1983)
35.10 B.D.O. Anderson, J.B. Moore: *Optimal Filtering* (Prentice Hall, Englewood Cliffs 1979)
35.11 J. Manyika, H.F. Durrant-Whyte: *Data Fusion and Sensor Management: An Information-Theoretic Approach* (Ellis Horwood, New York 1994)
35.12 S. Sukkarieh, E. Nettleton, J.H. Kim, M. Ridley, A. Goktogan, H. Durrant-Whyte: The ANSER project: Data fusion across multiple uninhabited air vehicles, Int. J. Robotics Res. **22**(7), 505–539 (2003)
35.13 R.E. Moore: *Interval Analysis* (Prentice Hall, Englewood Cliffs 1966)
35.14 D. Dubois, H. Prade: *Fuzzy Sets and Systems: Theory and Applications* (Academic, New York 1980)
35.15 S. Blackman, R. Popoli: *Design and Analysis of Modern Tracking Systems* (Artec House, Boston 1999)
35.16 D. Pagac, E.M. Nebot, H. Durrant-Whyte: An evidential approach to map-building for autonomous vehicles, IEEE Trans. Robotics Autom. **14**(4), 623–629 (1998)
35.17 D. Hall, J. Llinas: *Handbook of Multisensor Data Fusion* (CRC, Boca Raton 2001)
35.18 E.L. Waltz, J. Llinas: *Sensor Fusion* (Artec House, Boston 1991)
35.19 M. Kam, Z. Zhu, P. Kalata: Sensor fusion for mobile robot navigation, Proceedings IEEE **85**, 108–119 (1997)
35.20 H. Carvalho, W. Heinzelman, A. Murphy, C. Coelho: A general data fusion architecture, Proc. 6th Int. Conf. Inf. Fusion, Cairns (2003)
35.21 A. Makarenko: A Decentralized Architecture for Active Sensor Networks, Ph.D. Thesis (University of Sydney, Sydney 2004)
35.22 M. Dekhil, T. Henderson: Instrumented logical sensors systems, Int. J. Robotics Res. **17**(4), 402–417 (1998)
35.23 J.A. Profeta: Safety-critical systems built with COTS, Computer **29**(11), 54–60 (1996)
35.24 H. Hu, J.M. Brady, F. Du, P. Probert: Distributed real-time control of a mobile robot, J. Intell. Autom. Soft Comput. **1**(1), 63–83 (1995)
35.25 S.A. Schneider, V. Chen, G. Pardo: ControlShell: A real-time software framework, AIAA Conf. Intell. Robotics Field Fact. Serv. Space (1994)
35.26 D. Simon, B. Espiau, E. Castillo, K. Kapellos: Computer-aided design of a generic robot controller handling reactivity and real-time issues, IEEE Trans. Control Syst. Technol. **4**(1), 213–229 (1993)
35.27 C. Giraud, B. Jouvencel: Sensor selection in a fusion process: a fuzzy approach, Proc. IEEE Int. Conf. Multisens. Fusion Integr., Las Vegas (1994) pp. 599–606
35.28 R. Kapur, T.W. Williams, E.F. Miller: System testing and reliability techniques for avoiding failure, Computer **29**(11), 28–30 (1996)
35.29 K.H. Kim, C. Subbaraman: Fault-tolerant real-time objects, Communication ACM **40**(1), 75–82 (1997)
35.30 D.B. Stewart, P.K. Khosla: Mechanisms for detecting and handling timing errors, Communication ACM **40**(1), 87–93 (1997)
35.31 G. Weller, F. Groen, L. Hertzberger: A sensor processing model incorporating error detection and recovery. In: *Traditional and Non-Traditional Robotic Sensors*, ed. by T. Henderson (Springer, Berlin, Heidelberg 1990) pp. 351–363
35.32 R.R. Brooks, S. Iyengar: *Averaging Algorithm for Multi-Dimensional Redundant Sensor Arrays: Resolving Sensor Inconsistencies*, Tech. Rep. (Louisiana State University, Baton Rouge 1993)
35.33 T.C. Henderson, M. Dekhil: *Visual Target Based Wall Pose Estimation*, Tech. Rep. UUCS-97-010 (University of Utah, Salt Lake City 1997)

Avidbots Ex1016
Page 924 of 2154

35.34 S. Iyengar, D. Jayasimha, D. Nadig: A versatile architecture for the distributed sensor integration problem, Computer **43**, 175–185 (1994)
35.35 D. Nadig, S. Iyengar, D. Jayasimha: A new architecture for distributed sensor integration, Proc. IEEE Southeastcon (1993)
35.36 L. Prasad, S. Iyengar, R.L. Kashyap, R.N. Madan: Functional characterization of fault tolerant integration in distributed sensor networks, IEEE Trans. Syst. Man Cybern. **25**, 1082–1087 (1991)
35.37 L. Prasad, S. Iyengar, R. Rao, R. Kashyap: Fault-tolerence sensor integration using multiresolution decomposition, Am. Phys. Soc. **49**(4), 3452–3461 (1994)
35.38 H.F. Durrant-Whyte: *Integration, Coordination, and Control of Multi-Sensor Robot Systems* (Kluwer, Boston 1987)
35.39 F. Groen, P. Antonissen, G. Weller: Model based robot vision, IEEE Instrum. Meas. Technol. Conf. (1993) pp. 584–588
35.40 R. Joshi, A.C. Sanderson: Model-based multisensor data fusion: A minimal representation approach, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (1994)
35.41 A.J. Briggs, B.R. Donald: Automatic sensor configuration for task-directed planning, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (1994) pp. 1345–1350
35.42 G. Hager: *Task Directed Sensor Fusion and Planning* (Kluwer, Boston 1990)
35.43 G. Hager, M. Mintz: Computational methods for task-directed sensor data fusion and sensor planning, Int. J. Robotics Res. **10**(4), 285–313 (1991)
35.44 B. Donald: On information invariants in robotics, Artif. Intell. **72**, 217–304 (1995)
35.45 V. Braitenberg: *Vehicles: Experiments in Synthetic Psychology* (MIT Press, Cambridge 1984)
35.46 R.A. Brooks: A robust layered control system for a mobile robot, IEEE Trans. Robotics Autom. **2**(1), 14–23 (1986)
35.47 K.P. Valavanis, A.L. Nelson, L. Doitsidis, M. Long, R.R. Murphy: *Validation of a Distributed Field Robot Architecture Integrated with a Matlab Based Control Theoretic Environment: A Case Study of Fuzzy Logic Based Robot Navigation*, CRASAR Tech. Rep. 25 (University of South Florida, Tampa 2004)
35.48 R.R. Murphy: *Introduction to AI Robotics* (MIT Press, Cambridge 2000)
35.49 S. Lee: Sensor fusion and planning with perception-action network, Proc. IEEE Conf. Multisens. Fusion Integr. Intell. Syst., Washington (1996)
35.50 S. Lee, S. Ro: Uncertainty self-management with perception net based geometric data fusion, Proc. IEEE Conf. Robotics Autom. (ICRA), Albuquerque (1997)
35.51 B.A. Draper, A.R. Hanson, S. Buluswar, E.M. Riseman: Information acquisition and fusion in the mobile perception laboratory, Proc. SPIE Sens. Fusion VI (1993)
35.52 S.S. Shafer, A. Stentz, C.E. Thorpe: An architecture for sensor fusion in a mobile robot, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (1986) pp. 2002–2007
35.53 S. Nagata, M. Sekiguchi, K. Asakawa: Mobile robot control by a structured hierarchical neural network, IEEE Control Syst. Mag. **10**(3), 69–76 (1990)
35.54 M. Pachter, P. Chandler: Challenges of autonomous control, IEEE Control Syst. Mag. **18**(4), 92–97 (1998)
35.55 R. Joshi, A.C. Sanderson: *Multisensor Fusion* (World Scientific, Singapore 1999)
35.56 V. Berge-Cherfaoui, B. Vachon: Dynamic configuration of mobile robot perceptual system, Proc. IEEE Conf. Multisens. Fusion Integr. Intell. Syst., Las Vegas (1994)
35.57 V. Clement, G. Giraudon, S. Houzelle, F. Sandakly: *Interpretation of Remotely Sensed Images in a Context of Multisensor Fusion Using a Multi-Specialist Architecture*, Rapp. Rech.: No. 1768 (INRIA, Sophia-Antipolis 1992)
35.58 J. Albus: RCS: A cognitive architecture for intelligent multi-agent systems, Proc. IFAC Symp. Intell. Auton. Veh., Lisbon (2004)
35.59 R. Camden, B. Bodt, S. Schipani, J. Bornstein, R. Phelps, T. Runyon, F. French: *Autonomous Mobility Technology Assessment*, Interim Rep., ARL-MR 565 (Army Research Laboratory, Washington 2003)
35.60 T. Queeney, E. Woods: A generic architecture for real-time multisensor fusion tracking algorithm development and evaluation, Proc. SPIE Sens. Fusion VII, Vol. 2355 (1994) pp. 33–42
35.61 T. Henderson, E. Shilcrat: Logical sensor systems, J. Robotics Syst. **1**(2), 169–193 (1984)
35.62 T. Henderson, C. Hansen, B. Bhanu: The specification of distributed sensing and control, J. Robotics Syst. **2**(4), 387–396 (1985)
35.63 J.D. Elliott: Multisensor Fusion within an Encapsulated Logical Device Architecture, Master's Thesis (University of Waterloo, Waterloo 2001)
35.64 M.D. Naish: Elsa: An Intelligent Multisensor Integration Architecture for Industrial Grading Tasks, Master's Thesis (University of Western Ontario, London 1998)
35.65 A. Makarenko, A. Brooks, S. Williams, H. Durrant-Whyte, B. Grocholsky: A decentralized architecture for active sensor networks, Proc. IEEE Int. Conf. Robotics Autom. (ICRA), New Orleans (2004) pp. 1097–1102
35.66 B. Grocholsky, A. Makarenko, H. Durrant-Whyte: Information-theoretic coordinated control of multiple sensor platforms, Proc. IEEE Int. Conf. Robotics Autom. (ICRA), Taipei (2003) pp. 1521–1527
35.67 R. Gregor, M. Lützeler, M. Pellkofer, K.-H. Siedersberger, E. Dickmanns: EMS-Vision: A perceptual system for autonomous vehicles, IEEE Trans. Intell. Transp. Syst. **3**(1), 48–59 (2002)
35.68 B. Rao, H. Durrant-Whyte, A. Sheen: A fully decentralized multi-sensor system for tracking and surveillance, Int. J. Robotics Res. **12**(1), 20–44 (1993)
35.69 B. Upcroft: Non-gaussian state estimation in an outdoor decentralised sensor network, Proc. IEEE Conf. Decis. Control (CDC) (2006)
35.70 S. Kumar, F. Ramos, B. Upcroft, H. Durrant-Whyte: A statistical framework for natural feature repre-

Avidbots Ex1016
Page 925 of 2154

sentation, Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS), Edmonton (2005) pp. 1–6
35.71 S. Gao, Y. Zhong, W. Li: Random weighting method for multisensor data fusion, IEEE Sens. J. **11**(9), 1955–1961 (2011)
35.72 M. Lhuillier: Incremental fusion of structure-from-motion and GPS using constrained bundle adjustments, IEEE Trans. Pattern Anal. Mach. Intell. **34**(12), 2489–2495 (2012)
35.73 S. Yu, L. Tranchevent, X. Liu, W. Glanzel, J. Suykens, B. DeMoor, Y. Moreau: Optimized data fusion for kernel k-means clustering, IEEE Trans. Pattern Anal. Mach. Intell. **34**(5), 1031–1039 (2012)
35.74 K. Kolev: Fast joint estimation of silhouettes and dense 3D geometry from multiple images, IEEE Trans. Pattern Anal. Mach. Intell. **34**(3), 493–505 (2012)
35.75 C. Loy: Incremental activity modeling in multiple disjoint cameras, IEEE Trans. Pattern Anal. Mach. Intell. **34**(9), 1799–1813 (2012)
35.76 N. Poh, J. Kittler: A unified framework for biometric expert fusion incorporating quality measures, IEEE Trans. Pattern Anal. Mach. Intell. **34**(1), 3–18 (2012)
35.77 M.-F. Weng, Y.-Y. Chuang: Cross-domain multicue fusion for concept-based video indexing, IEEE Trans. Pattern Anal. Mach. Intell. **34**(10), 1927–1941 (2012)
35.78 M. Hwangbo, J.-S. Kim, T. Kanade: Gyro-aided feature tracking for a moving camera: fusion, auto-calibration and GPU implementation, Intl. J. Robotics Res. **30**(14), 1755–1774 (2011)
35.79 H. Seraji, N. Serrano: A multisensor decision fusion system for terrain safety assessment, IEEE Trans. Robotics **25**(1), 99–108 (2009)
35.80 H. Himberg, Y. Motai, A. Bradley: Interpolation volume calibration: a multisensor calibration technique for electromagnetic trackers, IEEE Trans. Robotics **28**(5), 1120–1130 (2012)
35.81 K. Zhou, S.I. Roumeliotis: Optimal motion strategies for range-Only constrained multisensor target tracking, IEEE Trans. Robotics **24**(5), 1168–1185 (2008)
35.82 N.R. Ahmed, E.M. Sample, M. Campbell: Bayesian multicategorical soft data fusion for human–robot collaboration, IEEE Trans. Robotics **PP**(99), 1–18 (2012)
35.83 H. Frigui, L. Zhang, P.D. Gader: Context-dependent multisensor fusion and its application to land mine detection, IEEE Trans. Geosci. Remote Sens. **48**(6), 2528–2543 (2010)
35.84 J.G. Garcia, A. Robertson, J.G. Ortega, R. Johansson: Sensor fusion for compliant robot motion control, IEEE Trans. Robotics **24**(2), 430–441 (2008)
35.85 R. Heliot, B. Espiau: Multisensor input for CPG-based sensory–motor coordination, IEEE Trans. Robotics **24**(1), 191–195 (2008)
35.86 S. Liu, R.X. Gao, D. John, J.W. Staudenmayer, P.S. Freedson: Multisensor data fusion for physical activity assessment, IEEE Trans. Bio-Med. Eng. **59**(3), 687–696 (2012)
35.87 A. Martinelli: Vision and IMU data fusion: closed-Form solutions for attitude, speed, absolute scale, and bias determination, IEEE Trans. Robotics **28**(1), 44–60 (2012)

Avidbots Ex1016
Page 926 of 2154

Multimedia Contents



Part D Manipulation and Interfaces

Ed. by Makoto Kaneko

36 Motion for Manipulation Tasks
James Kuffner, Pittsburgh, USA
Jing Xiao, Charlotte, USA

37 Contact Modeling and Manipulation
Imin Kao, Stony Brook, USA
Kevin M. Lynch, Evanston, USA
Joel W. Burdick, Pasadena, USA

38 Grasping
Domenico Prattichizzo, Siena, Italy
Jeffrey C. Trinkle, Troy, USA

39 Cooperative Manipulation
Fabrizio Caccavale, Potenza, Italy
Masaru Uchiyama, Sendai, Japan

40 Mobility and Manipulation
Oliver Brock, Berlin, Germany
Jaeheung Park, Suwon, Korea
Marc Toussaint, Stuttgart, Germany

41 Active Manipulation for Perception
Anna Petrovskaya, Stanford, USA
Kaijen Hsiao, Palo Alto, USA

42 Haptics
Blake Hannaford, Seattle, USA
Allison M. Okamura, Stanford, USA

43 Telerobotics
Günter Niemeyer, Glendale, USA
Carsten Preusche, Wessling, Germany
Stefano Stramigioli, Enschede, The Netherlands
Dongjun Lee, Seoul, Korea

44 Networked Robots
Dezhen Song, College Station, USA
Ken Goldberg, Berkeley, USA
Nak-Young Chong, Ishikawa, Japan

Part D, Manipulation and Interfaces, is separated into two subparts; the first half is concerned with manipulation where frameworks of modeling, motion planning, and control of grasp and manipulation of an object are addressed, and the second half is concerned with interfaces where physical human–robot interactions are handled. Humans can achieve grasping and manipulation of an object dexterously through hand–arm coordination. An optimum control skill for such a redundant system is naturally and gradually acquired through experience in our daily life. Especially, fingers play an important role for expressing human dexterity. Without dexterous fingers, it is hard for us to handle any daily tool, such as a pencil, keyboard, cup, knife, or fork. This dexterity is supported with active and passive compliance as well as the multiple sensory organs existing at the fingertip. Such dexterous manipulation enables us to clearly differentiate humans from other animals. Thus, manipulation is one of the most important functions for humans. We eventually acquired the current shape of finger, the sensory organs, and skill for manipulation, through a long history of evolution, over more than six million years. While humans and robots are largely different in terms of actuators, sensors, and mechanisms, achieving dexterous manipulation like that of a human in a robot is a challenging subject in robotics. As we overview current robot technology, however, we observe that the dexterity of robots is still far behind that of humans. With this overview, we now provide a brief synopsis of each chapter in the first half of Part D.

Chapter 36, Motion for Manipulation Tasks, discusses algorithms that generate motion for manipulation tasks at the arm level, especially in an environment, by using the configuration space formalism. While in previous chapters (6 and 7) the focus was on specific algorithmic techniques for robot motion, this chapter is focused on a specific application for robot manipulation. The important example of assembly motion is discussed through the analysis of contact states and compliant motion control.

Chapter 37, Contact Modeling and Manipulation, provides the contact modeling on rigid contact with and without friction, and also the modeling on soft contact, such as elastic and viscoelastic contact interfaces. Kinematics and mechanics with friction are precisely handled under rigid-body contact. The selection matrix $\mathbf{H}$ is introduced to understand the force and velocity constraints at the contact interface. Pushing manipulation is also addressed by using the concept of the friction limit surface.

Chapter 38, Grasping, discusses based on the closure property, grasping with many examples, supposing multifingered robotic hands. A strong constraint for grasping is the unilateral characteristic,where a fingertip can push but not pull an object through a contact point. The rigid-body model is further extended for considering compliance in grasping with robotic hands having a low number of degrees of freedom (DOFs). Kinematics and closure issues are also addressed under this unilateral constraint.

Chapter 39, Cooperative Manipulators, addresses the strategies for controlling both the motion of cooperative system and the interaction forces between the manipulators and the grasped object when two manipulator arms firmly grasp a common object. It should be noted that this chapter allows the bilateral constraint where both directional force and moment are permissible. A general cooperative task-space formulation is introduced for different classes of multiarm robotic systems.

Chapter 40, Mobile Manipulation, focuses on research conducted on an experimental platform that combines capabilities in mobility and manipulation. Furthermore, it involves the interaction between the robots and real-world environment, unstructured environments. The main research objective here is to maximize task generality of autonomous robotic systems, under minimizing the dependence on task-specific, hard-coded, or narrowly-relevant information.

Chapter 41, Active Manipulation for Perception, covers perceptual methods in which manipulation is an integral part of perception. There are advantages to use manipulation rather than vision for perception. For example, manipulation can be used to sense in poor-visibility conditions, and also to determine properties that require physical interaction. This chapter includes the methods that have been developed for inference, planning, recognition, and modeling in sensing-via-manipulation approaches are cover.

Without dexterity like that of humans, future robots will not be able to work instead of humans in environments where human cannot enter. In this sense, the implementation of dexterity into robots is one of the highlights of future robot design. Chapters 36–41 provide a good hint for enhancing dexterity for robots.

The second half of Part D addresses interfaces where humans control a robot or multiple robots through direct or indirect contact with robot(s).We now

provide a brief synopsis of each chapter in the second half of Part D.

Chapter 42, Haptics, discusses robotics devices that allow human operators to experience the sense of touch in remote or virtual environment. Two classes of force feedback haptic devices are discussed. One is an admittance device that senses the force applied by the operator and constrains the operator's position to match the appropriate deflection of a simulated object or surface; the other is an impedance haptic device that senses the position of the operator and then applies a force vector to the operator according to the computed behavior of the simulated object or surface. Haptic rendering from real time three-dimensional (3-D) image is also introduced in this chapter.

Chapter 43, Telerobotics, starts with a discussion on the classification of three different concepts: direct control where all slave motions are directly controlled by the user via the master interface, shared control where task execution is shared between direct control and local sensory control, and supervisory control where the user and slave are connected loosely with strong local autonomy. Various control issues such as lossy communication with Internet and operation with mobile robots, are also addressed.

Chapter 44, Networked Robots, focuses on the framework of computer networks which offer extensive computing, memory, and other resources that can dramatically improve performance. The chapter covers a wide span from the history of networked robots as it evolves from teleoperation to cloud robotics to how to build a networked robot. The very recent progress on cloud robotics and potential topics for future research are included later in the chapter.

In Haptics (Chap. 42) direct contact between human and robot is made, while in both Telerobotics (Chap. 43) and Networked Robots (Chap. 44), an appropriate distance between the human and robot is kept. One of the main issues is how to maintain appropriate control performance of the system in the presence of humans or a time lag between humans and robots.