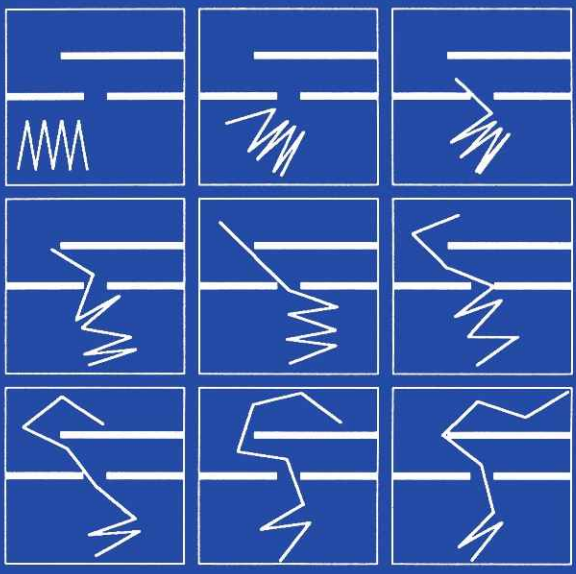
ROBOT MOTION PLANNING



Jean-Claude Latombe

TESLA EXHIBIT 1013, Page C1

ROBOT MOTION PLANNING

THE KLUWER INTERNATIONAL SERIES IN ENGINEERING AND COMPUTER SCIENCE

ROBOTICS: VISION, MANIPULATION AND SENSORS

Consulting Editor: **Takeo Kanade**

ROBOTIC GRASPING AND FINE MANIPULATION, M. Cutkosky
ISBN: 0-89838-200-9

SHADOWS AND SILHOUETTES IN COMPUTER VISION, S. Shafer
ISBN: 0-89838-167-3

PERCEPTUAL ORGANIZATION AND VISUAL RECOGNITION, D. Lowe
ISBN: 0-89838-172-X

ROBOT DYNAMICS ALGORITHMS, F. Featherstone
ISBN: 0-89838-230-0

THREE- DIMENSIONAL MACHINE VISION, T. Kanade (editor)
ISBN: 0-89838-188-6

KINEMATIC MODELING, IDENTIFICATION AND CONTROL OF *ROBOT MANIPULATORS,* H.W. Stone
ISBN: 0-89838-237-8

OBJECT RECOGNITION USING VISION AND TOUCH, P. Allen
ISBN: 0-89838-245-9

INTEGRATION, COORDINATION AND CONTROL OF MULTI-SENSOR ROBOT SYSTEMS, H.F. Durrant-Whyte
ISBN: 0-89838-247-5

MOTION UNDERSTANDING: Robot and Human Vision, W.N. Martin and J. K. Aggrawal (editors)
ISBN: 0-89838-258-0

BAYESIAN MODELING OF UNCERTAINTY IN LOW-LEVEL VISION, R. Szeliski
ISBN 0-7923-9039-3

VISION AND NAVIGATION: THE CMU NAVLAB, C. Thorpe (editor)
ISBN 0-7923-9068-7

TASK-DIRECTED SENSOR FUSION AND PLANNING: A Computational Approach, G. D. Hager
ISBN: 0-7923-9108-X

COMPUTER ANALYSIS OF VISUAL TEXTURES, F. Tomita and S. Tsuji
ISBN: 0-7923-9114-4

DATA FUSION FOR SENSORY INFORMATION PROCESSING *SYSTEMS,* J. Clark and A. Yuille
ISBN: 0-7923-9120-9

PARALLEL ARCHITECTURES AND PARALLEL ALGORITHMS FOR INTEGRATED VISION SYSTEMS, A.N. Choudhary, J. H. Patel
ISBN: 0-7923-9078-4

Robot Motion Planning

Jean-Claude Latombe

Stanford University



SPRINGER SCIENCE+BUSINESS MEDIA, LLC

TESLA EXHIBIT 1013, Page iii

Library of Congress Cataloging-in-Publication Data

Latombe, Jean Claude.
Robot motion planning / Jean-Claude Latombe.
p. cm. - (The Kluwer international series in engineering and computer science ; SECS 0124)
Includes bibliographical references and index.
ISBN 978-0-7923-9206-4 ISBN 978-1-4615-4022-9 (eBook)
DOI 10.1007/978-1-4615-4022-9
1. Robots - Motion. I. Title. II. Series.
TJ211.4.L38 1991
629.8'92 - dc20

90-49962
CIP

Copyright 1991 by Springer Science+Business Media New York
Originally published by Kluwer Academic Publishers, New York in 1991

Second printing, 1991.

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, mechanical, photo-copying, recording, or otherwise, without the prior written permission of the publisher, Springer Science+Business Media, LLC.

Printed on acid-free paper.

To Claudine, Laurence, and Emmanuel

Contents

	Preface	ix
	Notations	xvii
Chapter 1	Introduction and Overview	1
Chapter 2	Configuration Space of a Rigid Object	58
Chapter 3	Obstacles in Configuration Space	105
Chapter 4	Roadmap Methods	153
Chapter 5	Exact Cell Decomposition	200
Chapter 6	Approximate Cell Decomposition	248
Chapter 7	Potential Field Methods	295
Chapter 8	Multiple Moving Objects	356
Chapter 9	Kinematic Constraints	403
Chapter 10	Dealing with Uncertainty	452
Chapter 11	Movable Objects	533
	Prospects	587
Appendix A	Basic Mathematics	590
Appendix B	Computational Complexity	599
Appendix C	Graph Searching	603
Appendix D	Sweep-Line Algorithm	609
	References	615
	Index	643

Notations

The following conventions and notations are used throughout the book. Most of them are introduced in Chapter 1.

1. The robot is called $\mathcal{A}$. If there are several robots, they are called $\mathcal{A}_i$ $(i = 1, 2, ...)$.

2. The robots' workspace is denoted by $\mathcal{W}$ and is modeled as the Euclidean space $\mathbf{R}^N$, with $N = 2$ or 3. ($\mathbf{R}$ is the set of the real numbers.)

3. A Cartesian frame $\mathcal{F}_\mathcal{A}$ is attached to $\mathcal{A}$. Another Cartesian frame $\mathcal{F}_\mathcal{W}$ is attached to $\mathcal{W}$. The origin of $\mathcal{F}_\mathcal{A}$ (resp. $\mathcal{F}_\mathcal{W}$) is denoted by $O_\mathcal{A}$ (resp. $O_\mathcal{W}$).

4. The obstacles in $\mathcal{W}$ are denoted by $\mathcal{B}_i$ $(i = 1, 2, ...)$. The symbol $\mathcal{B}$ sometimes denotes a particular obstacle. More often, it denotes the union of all the obstacles (called the obstacle region).

5. The configuration space of a robot $\mathcal{A}$ is denoted by $\mathcal{C}$. It is a manifold of dimension m. An element of $\mathcal{C}$ (i.e. a configuration) is denoted by $\mathbf{q}$. The region of the workspace occupied by the robot $\mathcal{A}$ at configuration $\mathbf{q}$ is denoted by $\mathcal{A}(\mathbf{q})$.

6. An obstacle $\mathcal{B}$ in the workspace maps in configuration space to a region called C-obstacle and denoted by $\mathcal{CB}$.

7. The free space in configuration space is denoted by $\mathcal{C}_{free}$; the contact space by $\mathcal{C}_{contact}$. $\mathcal{C}_{valid} = \mathcal{C}_{free} \cup \mathcal{C}_{contact}$ is the valid space.

8. A path is denoted by τ. It is usually a function of a parameter denoted by s that takes its values in the interval $[0, 1]$. When τ is expressed as a function of time, it is called a trajectory.

9. A vector is denoted by a symbol with an arrow on top of it, e.g. $\vec{v}$. Its modulus is written $\|\vec{v}\|$. The inner product of two vectors $\vec{v}_1$ and $\vec{v}_2$ is denoted by $< \vec{v}_1, \vec{v}_2 >$ or $\vec{v}_1 \cdot \vec{v}_2$. The outer product of $\vec{v}_1$ and $\vec{v}_2$ is denoted by $\vec{v}_1 \wedge \vec{v}_2$. The angle between $\vec{v}_1$ and $\vec{v}_2$ is written $angle(\vec{v}_1, \vec{v}_2)$.

10. The distance between two points x and y of a Euclidean space is denoted by $\|x - y\|$.

11. An interval in $\mathbf{R}$ bounded by a and b is denoted by $[a, b]$ if it is closed at both ends and by (a, b) if it is open at both ends. It is denoted by $[a, b)$ or $(a, b]$ if it is closed at one end and open at the other.

12. Let a be a real number. $sign(a)$ denotes the "sign" of a; it is $+1$ if $a > 0$, -1 if $a < 0$, and 0 otherwise.

13. The symbols $\oplus$ and $\ominus$ are the Minkowski operators for affine set addition and subtraction, respectively.

14. Let E be a topological space and F be a subset of it. $cl(F)$, $int(F)$, and $\partial(F)$ denote the closure, the interior, and the boundary of F, respectively. $E \backslash F$ denotes the complement of F in E.

15. Most of the time, calligraphic letters, e.g. $\mathcal{S}$, denote sets.

16. Symbols in typewriter type style are used to denote boolean expressions or predicates, e.g. `CB`, `Achieve`. The symbols `true` and `false` are the logical values "true" and "false".

ROBOT MOTION PLANNING

Chapter 1

Introduction and Overview

A robot is a versatile mechanical device — for example, a manipulator arm, a multi-joint multi-fingered hand, a wheeled or legged vehicle, a free-flying platform, or a combination of these — equipped with actuators and sensors under the control of a computing system. It operates in a workspace within the real world. This workspace is populated by physical objects and is subject to the laws of nature. The robot performs tasks by executing motions in the workspace.

In this book we are interested in giving the robot the capability of planning its own motions, i.e. deciding automatically what motions to execute in order to achieve a task specified by initial and goal spatial arrangements of physical objects. Creating autonomous robots is a major undertaking in Robotics. It definitively requires that the ability to plan motions automatically be developed. Indeed, except in limited and carefully engineered environments, it is not realistic to anticipate and explicitly describe to the robot all the possible motions that it may have to execute in order to accomplish requested tasks. Even in those cases where such a description is feasible, it would certainly be useful to incorporate automatic motion planning tools in off-line robot programming

systems. This would allow the user to specify tasks more declaratively, by stating *what* he/she wants done rather than *how* to do it. As robots become more dexterous, the need for motion planning tools will become more critical.

A still widespread view is that motion planning is essentially some sort of simple collision checking. Motion planning is much more than that. As we will see in this chapter, it presents an unexpected variety of facets, the simplest of which raise provably difficult computational issues. In fact, the kind of operative intelligence that people use unconsciously to interact with their environment, which is needed for perception and motion planning, turns out to be extremely difficult to duplicate in a computer program.

This chapter is both an introduction to robot motion planning and an overview of the rest of the book.

We first illustrate the various aspects of motion planning (Section 1). This illustration motivates our next step, the definition of a "basic problem" which is in some way the simplest useful motion planning problem that we may investigate (Section 2). Using this problem, we present the concept of configuration space, which is a unifying concept throughout the book (Section 3) and we outline the principles of the main computational approaches to motion planning (Section 4). We then introduce the extensions of the basic problem which will be studied in greater detail later (Section 5).

An important issue in motion planning is the computational complexity of the algorithms. In the following chapters of this book, we will analyze the complexity of particular planning methods. However, it is useful to have some preliminary idea of the intrinsic complexity of the problems which we are addressing. During the last few years, significant results have been obtained on this topic. We review them in Section 6. We also analyze the impact of these results on the design of operational systems, by proposing various guidelines for simplifying problems and implementing planners of increased practical efficiency (Section 7).

Motion planning is one major problem in the creation of autonomous robots. It is not the only one. It interacts with other important problems, including real-time motion control, sensing and task planning. We briefly discuss these interactions in Section 8.

TESLA EXHIBIT 1013, Page 2

The final section of this chapter reviews some key publications (Section 9). It is an historical account of the development of the motion planning field from the late 60's to the present time. Of course, other publications not mentioned in that section have also brought significant contributions to the field. We will reference many of them later in the book.

1 Aspects of Motion Planning

Consider a robot arm whose task is to build an assembly from a set of individual mechanical parts. Assume that the task is described as both a set of models of the parts, including their geometry and perhaps some physical properties, and a sequence of spatial relations [Latombe, 1984]. Figure 1 illustrates this description with a simple construction task. The sequence determines the order in which the parts are to be assembled. Each relation describes the position of a new part with respect to the current subassembly. The automatic transformation of this sequence into a description of the robot motions to be executed is one possible application of the methods presented in the book.

Let us analyze the example in more detail. In order to assemble the parts according to the input relations, the robot has to successively grasp each part, transfer it to the current subassembly, and position it on the subassembly:

- **Grasping** the part requires the position of the robot's gripper on the object to be selected and a path to this position to be generated. The grasp position must be accessible; it must also be stable and robust enough to resist some external forces. Sometimes, a satisfactory grasping position requires the part to be grasped, put down, and re-grasped.

- **Transferring** the part requires the geometry of a path for the arm to be computed. Since the motion will typically be executed at high speed, the path should avoid any contact with obstacles. Due to imprecision in motion control, it may even have to guarantee some minimal clearance.

- **Positioning** the part on the subassembly (part mating) requires motion commands to be selected for achieving a goal with high precision in a very constrained space. In general, due to uncertainty, a single

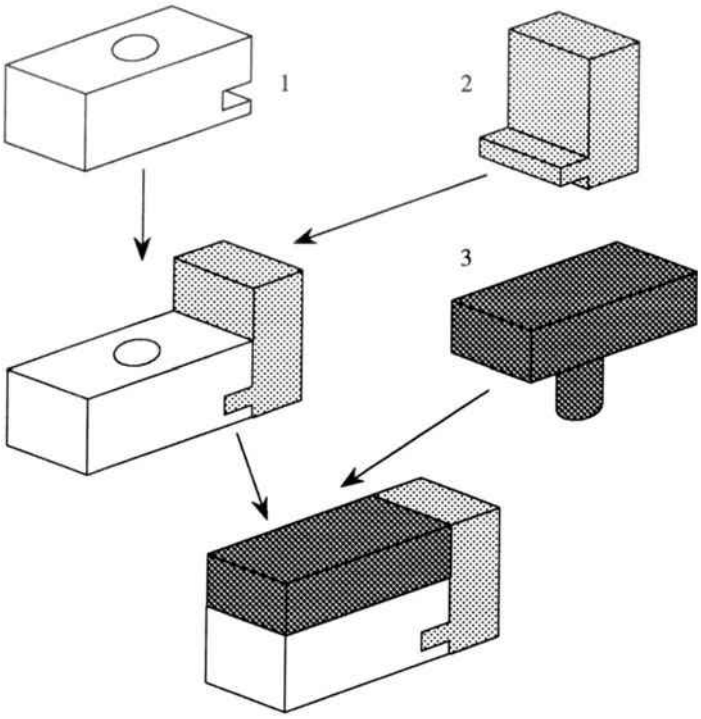


Figure 1. This simple construction task is defined by both the models of three constituent parts and a sequence of two spatial relations. Each relation may be input as a set of primitive relations among geometrical features, e.g. "Face A of Object 1 is against Face B of Object 2", "Shaft C of Object 3 fits Hole D of Object 1" [Ambler and Popplestone, 1975] [Mazer, 1983]. It may then be converted into a quantitative internal representation, e.g. the "homogeneous coordinate matrix" describing the transform between two Cartesian coordinate frames attached to the objects [Craig, 1986].

motion command is not enough; instead, a motion strategy involving several commands and using sensory inputs is necessary.

Prior knowledge of the spatial arrangement of the robot workspace is needed to plan these operations. For example, planning a collision-free path requires knowing where the obstacles are located. If complete knowledge is available (it may have been provided by the user or by a computer-aided design system, or obtained through sensing), motion

plans can be fully generated and then executed. If this is not the case, some interweaving of planning and execution is necessary. When uncertainty is well-bounded (this is usually the case during part mating), robust plans anticipating various contingencies may be suitable.

The above example can easily be made more complicated. For instance, there might be more than one robot to achieve the task, or there might be mobile obstacles in the workspace. Then, some motion coordination is necessary, implying that motions are planned not just as geometrical paths, but as functions of time. A robot might also be equipped with a dexterous multi-joint multi-fingered hand allowing complex types of motions within the hand. The task might include some performance criterion to optimize, e.g. minimize the total amount of time needed to perform the task. Rather than specifying a sequence of spatial relations, the input description might only describe the spatial arrangement of the parts in the goal assembly, leaving to the planner the generation of the appropriate sequence.

The example stresses the fact that motion planning does not consist of a single, well-delimited problem. Instead, it looks more like a collection of many problems, which are more or less variants of each other. Although one may envision that a general formulation covering all possible cases could be ultimately established, such a formulation is certainly beyond our current understanding of the conceptual and computational issues underlying motion planning.

In order to organize the book, we will first consider a "basic motion planning problem" which deals with geometrical issues only, and then several extensions of this problem which involve more sophisticated geometrical and non-geometrical issues. The basic problem is defined in Section 2 and several extensions are presented in Section 5.

2 Basic Problem

The goal of defining a basic motion planning problem is to isolate some central issues and investigate them in depth before considering additional difficulties.

In the basic problem, we assume that the robot is the only moving object in the workspace and we ignore the dynamic properties of the robot,

thus avoiding temporal issues. We also restrict motions to non-contact motions, so that the issues related to the mechanical interaction between two physical objects in contact can be ignored. These assumptions essentially transform the "physical" motion planning problem into a purely geometrical path planning problem. We simplify even further the geometrical issues by assuming that the robot is a single rigid object, i.e. an object whose points are fixed with respect to each other. The motions of this object are only constrained by the obstacles.

The **basic motion planning problem** resulting from these simplifications is the following:

> Let $\mathcal{A}$ be a single rigid object — the *robot* — moving in a Euclidean space $\mathcal{W}$, called *workspace*, represented as $\mathbf{R}^N$, with $N = 2$ or 3.
>
> Let $\mathcal{B}_1, ..., \mathcal{B}_q$ be fixed rigid objects distributed in $\mathcal{W}$. The $\mathcal{B}_i$'s are called *obstacles.*
>
> Assume that both the geometry of $\mathcal{A}, \mathcal{B}_1, ..., \mathcal{B}_q$ and the locations of the $\mathcal{B}_i$'s in $\mathcal{W}$ are accurately known. Assume further that no kinematic constraints limit the motions of $\mathcal{A}$ (we say that $\mathcal{A}$ is a *free-flying object*).
>
> The problem is: Given an initial position and orientation and a goal position and orientation of $\mathcal{A}$ in $\mathcal{W}$, generate a *path* τ specifying a continuous sequence of positions and orientations of $\mathcal{A}$ avoiding contact with the $\mathcal{B}_i$'s, starting at the initial position and orientation, and terminating at the goal position and orientation. Report failure if no such path exists.

Figure 2 illustrates two instances of the basic motion planning problem. It shows the paths of a rectangle (Figure 2.a) and an L-shaped polygon (Figure 2.b) among polygonal obstacles in a two-dimensional workspace.

Of course, the basic problem is somewhat oversimplified. We will see that it is nonetheless a difficult problem and that several of its solutions have straightforward extensions to more involved problems. Furthermore, solving it is also of practical interest. For instance, some mobile robot navigation problems can be realistically expressed as instances of the basic problem.

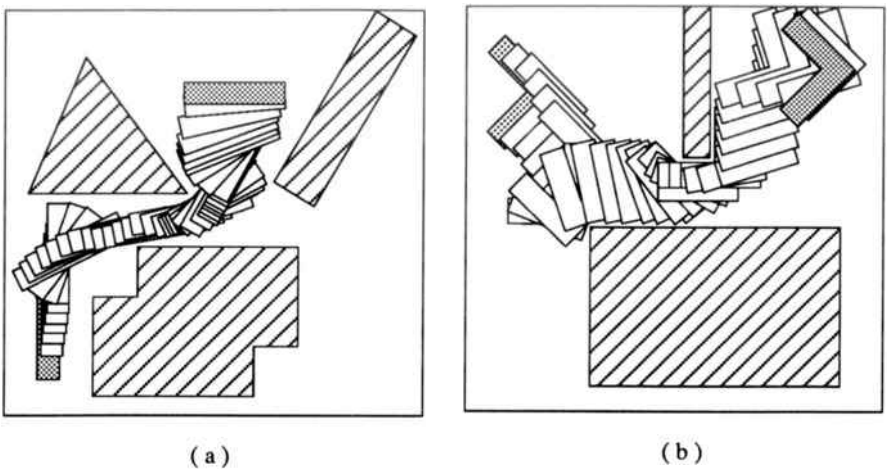


Figure 2. This figure illustrates the basic motion planning problem with two robots — a rectangle in Figure a and an L-shaped polygon in Figure b. The contact-free paths are displayed as discrete sequences of positions and orientations of the robot.

Although intuitively simple, some aspects of the above definition are still too informal and must be made more precise. For instance, the concept of a "free-flying" object is probably not very clear at this point. One possible formal statement of the problem is known as the *configuration space formulation.*

3 Configuration Space Formulation

The underlying idea of configuration space is to represent the robot as a point in an appropriate space — the robot's configuration space — and to map the obstacles in this space. This mapping transforms the problem of planning the motion of a dimensioned object into the problem of planning the motion of a point. It makes the constraints on the motions of the robot more explicit.

3.1 Notion of Configuration Space

Consider the basic problem. Let the robot $\mathcal{A}$ (at a certain position and orientation) be described as a compact (i.e. closed and bounded) subset

of $\mathcal{W} = \mathbf{R}^N$, $N = 2$ or 3, and the obstacles $\mathcal{B}_1, ..., \mathcal{B}_q$ be closed subsets of $\mathcal{W}$. In addition, let $\mathcal{F}_\mathcal{A}$ and $\mathcal{F}_\mathcal{W}$ be Cartesian frames embedded in $\mathcal{A}$ and $\mathcal{W}$, respectively. $\mathcal{F}_\mathcal{A}$ is a moving frame, while $\mathcal{F}_\mathcal{W}$ is a fixed one. By definition, since $\mathcal{A}$ is rigid, every point a of $\mathcal{A}$ has a fixed position with respect to $\mathcal{F}_\mathcal{A}$. But a's position in $\mathcal{W}$ depends on the position and orientation of $\mathcal{F}_\mathcal{A}$ relative to $\mathcal{F}_\mathcal{W}$. Since the $\mathcal{B}_i$'s are both rigid and fixed in $\mathcal{W}$, every point of $\mathcal{B}_i$, for all $i \in [1, q]$, has a fixed position with respect to $\mathcal{F}_\mathcal{W}$.

A configuration of an arbitrary object is a specification of the position of every point in this object relative to a fixed reference frame [Arnold, 1978]. Therefore, a **configuration q** of $\mathcal{A}$ is a specification of the position $\mathcal{T}$ and orientation Θ of $\mathcal{F}_\mathcal{A}$ with respect to $\mathcal{F}_\mathcal{W}$. The **configuration space** of $\mathcal{A}$ is the space $\mathcal{C}$ of all the configurations of $\mathcal{A}$. The subset of $\mathcal{W}$ occupied by $\mathcal{A}$ at configuration $\mathbf{q}$ is denoted by $\mathcal{A}(\mathbf{q})$. In the same fashion, the point a on $\mathcal{A}$ at configuration $\mathbf{q}$ is denoted by $a(\mathbf{q})$ in $\mathcal{W}$.

One may describe a configuration by a list of real parameters. For example, the position $\mathcal{T}$ can be simply described by the vector of the N coordinates of the origin of $\mathcal{F}_\mathcal{A}$ in $\mathcal{F}_\mathcal{W}$. The orientation Θ can be described as the $N \times N$ matrix whose columns are the components, in $\mathcal{F}_\mathcal{W}$, of the unit vectors along the axes of $\mathcal{F}_\mathcal{A}$. Then, $\mathbf{q} = (\mathcal{T}, \Theta)$ is uniquely represented as a list of $N(N+1)$ parameters. Clearly, however, this representation is redundant, since the matrix describing Θ must have orthonormal columns and determinant $+1$. Therefore, $\mathcal{C}$ is only a subset of $\mathbf{R}^{N(N+1)}$.

One may be tempted to describe Θ by a list of parameters of minimal cardinality, i.e. one angle θ, if $N = 2$ (for example, the angle between the x-axes of $\mathcal{F}_\mathcal{W}$ and $\mathcal{F}_\mathcal{A}$), and three angles ϕ, θ and ψ, if $N = 3$ (for example, the Euler angles). However, such a minimal description is not injective, i.e. the same configuration may be described by different values of the angles. For example, in the two-dimensional case ($N = 2$), two angles whose difference is a multiple of 2π represent the same Θ. In this case, the difficulty can easily be fixed by restricting the values of the angle θ to the $[0, 2\pi)$ interval with modulo 2π arithmetic. In the three-dimensional case ($N = 3$), the difficulty is slightly harder to remedy.

Describing a configuration as a list of m independent parameters, with

$m = 3$ (if $N = 2$) and $m = 6$ (if $N = 3$), corresponds to representing $\mathcal{C}$ as $\mathbf{R}^m$. However, as we will see at greater length in Chapter 2, while $\mathcal{C}$ is locally "like" $\mathbf{R}^m$, it is globally different. This means that $\mathcal{C}$ can be decomposed into a finite union of p ($p > 1$) slightly overlapping "patches", each represented as a copy of $\mathbf{R}^m$ (called a *chart*). Hence, a configuration can locally be represented as a non-redundant list of m real parameters, but some administration is required to switch between representations when the robot's configuration moves from one patch to another. The number m is called the **dimension** of $\mathcal{C}$.

3.2 Notion of a Path

The notion of *continuity* is fundamental in the definition of a path. Its formalization requires us to define a topology in $\mathcal{C}$. One classical way to do so is to specify a distance function $d : \mathcal{C} \times \mathcal{C} \rightarrow \mathbf{R}$, and to let the topology in $\mathcal{C}$ be the metric topology induced by d. In order to match our intuition of the real-world physics, the distance between two configurations $\mathbf{q}$ and $\mathbf{q}'$ should decrease and tend toward zero when the regions $\mathcal{A}(\mathbf{q})$ and $\mathcal{A}(\mathbf{q}')$ get closer to each other and tend to coincide. A simple distance function that satisfies this condition is:

$$d(\mathbf{q}, \mathbf{q}') = \max_{a \in \mathcal{A}} \|a(\mathbf{q}) - a(\mathbf{q}')\|$$

where $\|x - x'\|$ denotes the Euclidean distance between any two points x and x' in $\mathbf{R}^N$. Throughout this book, the topology in $\mathcal{C}$ will be the topology induced by this distance.[1]

A **path** of $\mathcal{A}$ from the configuration $\mathbf{q}_{init}$ to the configuration $\mathbf{q}_{goal}$ is a continuous map:

$$\tau : [0, 1] \rightarrow \mathcal{C}$$

with:

$$\tau(0) = \mathbf{q}_{init} \quad \text{and} \quad \tau(1) = \mathbf{q}_{goal}.$$

$\mathbf{q}_{init}$ and $\mathbf{q}_{goal}$ are the **initial** and **goal** configurations of the path, respectively. The continuity of τ entails that for all $s_0 \in [0, 1]$, $\lim_{s \rightarrow s_0} \max_{a \in \mathcal{A}} \|a(\tau(s)) - a(\tau(s_0))\| = 0$, with s taking its values in $[0, 1]$. Saying that $\mathcal{A}$ is a "free-flying object" means that, in the absence of obstacles, any path defined as above is feasible.

[1] As is well-known, the notion of a topology is more primitive than the notion of a metric. In Chapter 2 we will give other metrics which all induce the same topology as d.

TESLA EXHIBIT 1013, Page 9

3.3 Obstacles in Configuration Space

The above definition of a path does not take obstacles into consideration. We now want to characterize the set of paths which are solutions to the basic problem when there are obstacles in the workspace. To that purpose, we map the obstacles in configuration space and we consider the subset of $\mathcal{C}$ that is made of contact-free configurations.

Every obstacle $\mathcal{B}_i$, $i = 1$ to q, in the workspace $\mathcal{W}$ maps in $\mathcal{C}$ to a region:

$$\mathcal{CB}_i = \{\mathbf{q} \in \mathcal{C} \;/\; \mathcal{A}(\mathbf{q}) \cap \mathcal{B}_i \neq \emptyset\}$$

which is called a **C-obstacle**. The union of all the C-obstacles:

$$\bigcup_{i=1}^{q} \mathcal{CB}_i$$

is called the **C-obstacle region**, and the set:

$$\mathcal{C}_{free} = \mathcal{C} \setminus \bigcup_{i=1}^{q} \mathcal{CB}_i = \{\mathbf{q} \in \mathcal{C} \;/\; \mathcal{A}(\mathbf{q}) \cap (\bigcup_{i=1}^{q} \mathcal{B}_i) = \emptyset\}$$

is called the **free space**. Any configuration in $\mathcal{C}_{free}$ is called a **free configuration**.

A **free path** between two free configurations $\mathbf{q}_{init}$ and $\mathbf{q}_{goal}$ is a continuous map $\tau : [0,1] \rightarrow \mathcal{C}_{free}$, with $\tau(0) = \mathbf{q}_{init}$ and $\tau(1) = \mathbf{q}_{goal}$. Two configurations belong to the same connected component of $\mathcal{C}_{free}$ if and only if they are connected by a free path.

Given an initial and a goal configuration, the basic motion planning problem is to generate a free path between the two configurations, if they belong to the same connected component of $\mathcal{C}_{free}$, and to report failure otherwise.

Several planning methods described in Chapters 4 through 7 generate semi-free paths, rather than free paths. A **semi-free path** is a continuous map $\tau : [0,1] \rightarrow cl(\mathcal{C}_{free})$, where $cl(\mathcal{C}_{free})$ denotes the closure of $\mathcal{C}_{free}$. Hence, as it moves along such a path, the robot may touch obstacles. Under simple assumptions stated in Chapter 2, any semi-free path τ can be transformed into a free path that can be made arbitrarily close to τ.

TESLA EXHIBIT 1013, Page 10

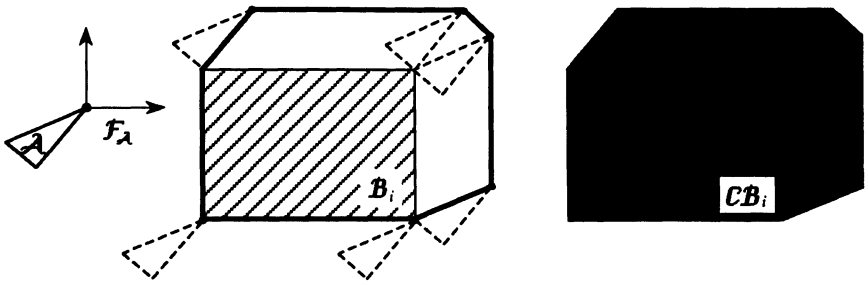


Figure 3. The robot $\mathcal{A}$ (a triangle) can translate freely in the plane at fixed orientation. Its configuration is represented as $\mathbf{q} = (x, y)$, the coordinates in $\mathcal{F}_\mathcal{W}$ of the vertex of $\mathcal{A}$ marked as a small circle (the origin of $\mathcal{F}_\mathcal{A}$). Hence, $\mathcal{A}$'s configuration space is $\mathcal{C} = \mathbf{R}^2$. The C-obstacle $\mathcal{CB}_i$ (shown dark) is obtained by "growing" the corresponding workspace obstacle $\mathcal{B}_i$ (a rectangle) by the shape of $\mathcal{A}$. Planning a motion of $\mathcal{A}$ relative to $\mathcal{B}_i$ is equivalent to planning a motion of the marked vertex of $\mathcal{A}$ relative to $\mathcal{CB}_i$.

3.4 Translational Case

When $\mathcal{A}$ consists of a single point (*point robot*), it is meaningless to talk about its orientation. Then, like $\mathcal{W}$, the configuration space of $\mathcal{A}$ is a copy of $\mathbf{R}^N$, and hence is a Euclidean space. The C-obstacles are identical to the obstacles in the workspace.

When $\mathcal{A}$ is a disc or when $\mathcal{A}$ is a dimensioned object allowed to translate freely (i.e. the origin of $\mathcal{F}_\mathcal{A}$ can follow any path in $\mathcal{W}$) without rotation (i.e. $\mathcal{F}_\mathcal{A}$ has a fixed orientation with respect to $\mathcal{F}_\mathcal{W}$), the configuration space $\mathcal{C}$ is also $\mathbf{R}^N$. But, in these cases, the C-obstacles are obtained by "growing" the obstacles by the shape of $\mathcal{A}$ as illustrated in Figure 3.

In the following, all these cases, where $\mathcal{C} = \mathbf{R}^N$, will be considered as particular cases of the basic motion planning problem. They will be convenient for introducing motion planning methods with simple examples.

4 Planning Approaches

There exists a large number of methods for solving the basic motion planning problem. Not all of them solve the problem in its full generality, however. For instance, some methods require the workspace to

be two-dimensional and the objects to be polygonal. Despite many external differences, the methods are based on few different general approaches: *roadmap*, *cell decomposition*, and *potential field*. We briefly introduce these approaches below and we illustrate them with simple examples in two-dimensional configuration spaces populated by polygonal C-obstacles. As shown in Figure 3, such configuration spaces correspond to the case where the robot $\mathcal{A}$ is a polygon (or a point) translating in a workspace containing polygonal obstacles. These approaches will be described in detail in Chapters 4 through 7.

4.1 Roadmap

The roadmap approach to path planning consists of capturing the connectivity of the robot's free space in a network of one-dimensional curves, called the *roadmap*, lying in the free space $\mathcal{C}_{free}$ or its closure $cl(\mathcal{C}_{free})$. Once a roadmap $\mathcal{R}$ has been constructed, it is used as a set of standardized paths. Path planning is thus reduced to connecting the initial and goal configurations to points in $\mathcal{R}$ and searching $\mathcal{R}$ for a path between these points. The constructed path, if any, is the concatenation of three subpaths: a subpath connecting the initial configuration to the roadmap, a subpath contained in the roadmap, and a subpath connecting the roadmap to the goal configuration.

Various methods based on this general idea have been proposed. They compute different types of roadmaps called *visibility graph*, *Voronoi diagram*, *freeway net* and *silhouette*. We illustrate two of them below.

The *visibility graph method* is one of the earliest path planning methods. It mainly applies to two-dimensional configuration spaces with a polygonal C-obstacle region. The visibility graph is the non-directed graph G whose nodes are the initial and goal configurations $\mathbf{q}_{init}$ and $\mathbf{q}_{goal}$, and all the C-obstacle vertices. The links of G are all the straight line segments connecting two nodes that do not intersect the interior of the C-obstacle region. Figure 4 shows an example of a visibility graph. The links of the subgraph of G that is restricted to the C-obstacle vertices determine the roadmap $\mathcal{R}$. The other links of G connect the initial and goal configurations to the roadmap. G can be searched for the shortest semi-free path between $\mathbf{q}_{init}$ and $\mathbf{q}_{goal}$ (according to the Euclidean metric in $\mathbf{R}^2$). This path, if it exists, is a polygonal line connecting $\mathbf{q}_{init}$ to $\mathbf{q}_{goal}$ through C-obstacle vertices.

TESLA EXHIBIT 1013, Page 12

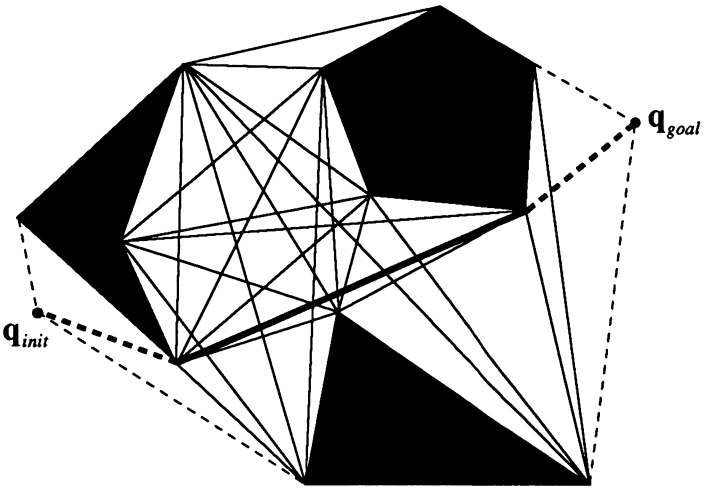


Figure 4. This figure shows the visibility graph in a two-dimensional configuration space with polygonal C-obstacles. The nodes of the graph are the initial and goal configurations and the C-obstacles' vertices. Two nodes are connected by a link if the straight line segment joining them does not intersect the C-obstacles' interior. (Hence, the links of the graph also include the edges of the C-obstacles.) The plain links connect C-obstacles' vertices. They are independent of the initial and goal configurations, and form the roadmap. The dashed links connect the initial and goal configurations to the roadmap. Any path in the visibility graph determines a semi-free path in configuration space. The graph contains the shortest semi-free path (according to the Euclidean metric in $\mathbf{R}^2$). This path is shown in both plain and dashed bold lines.

Another roadmap method, called *retraction*, consists of defining a continuous function of $\mathcal{C}_{free}$ onto a one-dimensional subset of itself (the roadmap) such that the restriction of this function to this subset is the identity map (in Topology such a function is called a "retraction"). In a two-dimensional configuration space, $\mathcal{C}_{free}$ is typically "retracted" onto its *Voronoi diagram*. This diagram is the set of all the free configurations whose minimal distance to the C-obstacle region $\mathcal{CB}$ is achieved with at least two points in the boundary of $\mathcal{CB}$ (Figure 5). The advantage of this diagram is that it yields free paths which tend to maximize the clearance between the robot and the obstacles. When the C-obstacles are polygons, the Voronoi diagram consists of straight and parabolic segments.

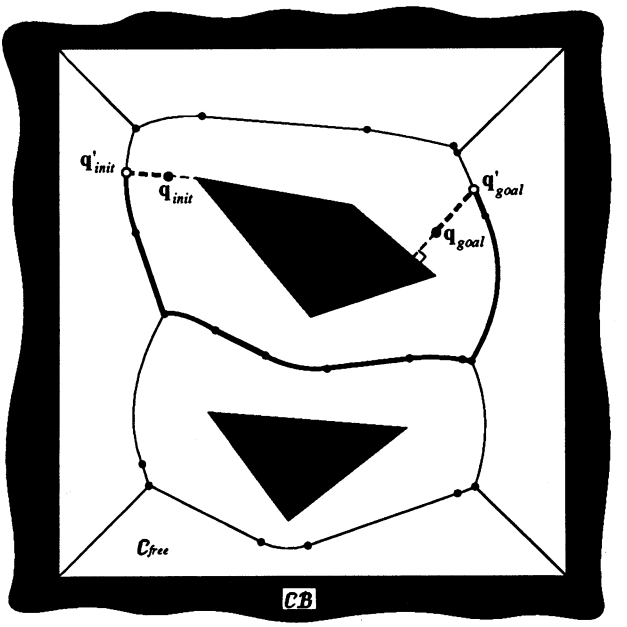


Figure 5. This figure shows the Voronoi diagram in a configuration space with a polygonal C-obstacle region. The free space in externally bounded by a rectangle (the boundary of a non-compact C-obstacle). The initial and goal configurations, $\mathbf{q}_{init}$ and $\mathbf{q}_{goal}$, are mapped in the Voronoi diagram to $\mathbf{q}'_{init}$ and $\mathbf{q}'_{goal}$, each by drawing the line along which its distance to the boundary of the C-obstacle region increases the fastest.

The initial and goal configurations are retracted in the diagram to $\mathbf{q}'_{init}$ and $\mathbf{q}'_{goal}$ as shown in Figure 5. A path is searched in the diagram between $\mathbf{q}'_{init}$ and $\mathbf{q}'_{goal}$. The free path between $\mathbf{q}_{init}$ and $\mathbf{q}_{goal}$ produced by the method, if any, consists of three subpaths: the straight path from $\mathbf{q}_{init}$ to $\mathbf{q}'_{init}$, a path in the diagram from $\mathbf{q}'_{init}$ to $\mathbf{q}'_{goal}$, and the straight path from $\mathbf{q}'_{goal}$ to $\mathbf{q}_{goal}$.

4.2 Cell Decomposition

Cell decomposition methods are perhaps the motion planning methods which have been the most extensively studied so far. They consist of decomposing the robot's free space into simple regions, called *cells*, such that a path between any two configurations in a cell can be easily gener-

ated. A non-directed graph representing the adjacency relation between the cells is then constructed and searched. This graph is called the *connectivity graph.* Its nodes are the cells extracted from the free space and two nodes are connected by a link if and only if the two corresponding cells are adjacent. The outcome of the search is a sequence of cells called a *channel.* A continuous free path can be computed from this sequence.

Cell decomposition methods can be broken down further into *exact* and *approximate* methods:

- Exact cell decomposition methods decompose the free space into cells whose union is exactly[2] the free space. The boundary of a cell corresponds to a criticality of some sort, i.e. a sudden change in the constraints applying to the motion of the robot.

- Approximate cell decomposition methods produce cells of predefined shape (e.g. rectangloids) whose union is strictly included in the free space. The boundary of a cell does not characterize a discontinuity of some sort and has no physical meaning.

Figures 6.1 and 6.2 illustrate an exact cell decomposition method in a two-dimensional configuration space. The free space is externally bounded by a polygon and internally bounded by three polygons. The free space is exactly decomposed into trapezoidal and triangular cells. These cells are built by drawing vertical rays from the C-obstacles' vertices. Two cells are adjacent if they share a portion of an edge of non-zero length. Once the connectivity graph has been constructed and a channel found, a free path is computed by connecting the initial configuration to the goal configuration through the midpoints of the intersections of every two successive cells.

Figure 7 illustrates an approximate cell decomposition method. The free space is externally bounded by a rectangle R and internally bounded by three polygons. The rectangle R is recursively decomposed into smaller rectangles. Each decomposition generates four identical new rectangles. (This type of decomposition is called a "quadtree" because it can be represented as a tree of degree 4.) At some level of resolution, only the

[2]The term "exact" refers to the input description of the motion planning problem, not to the actual physical problem, which can only be approximated in the input problem formulation.

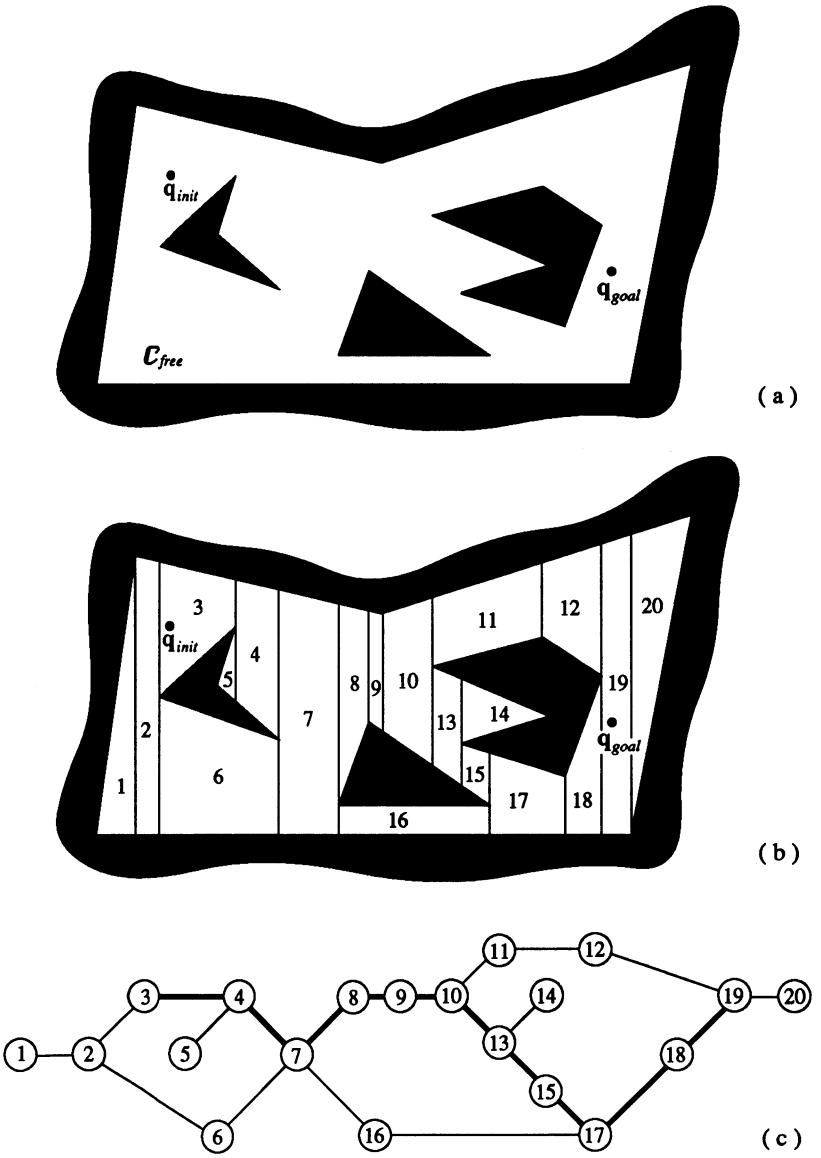


Figure 6.1. This figure and the next one illustrate an exact cell decomposition method. The free space is externally bounded by a polygon and internally bounded by three polygons (Figure a). The first step of the method consists of decomposing the free space into trapezoidal and triangular cells (Figure b). The second step is to construct the connectivity graph representing the adjacency relation between the cells (Figure c) and search this graph for a path (thick lines).

TESLA EXHIBIT 1013, Page 16

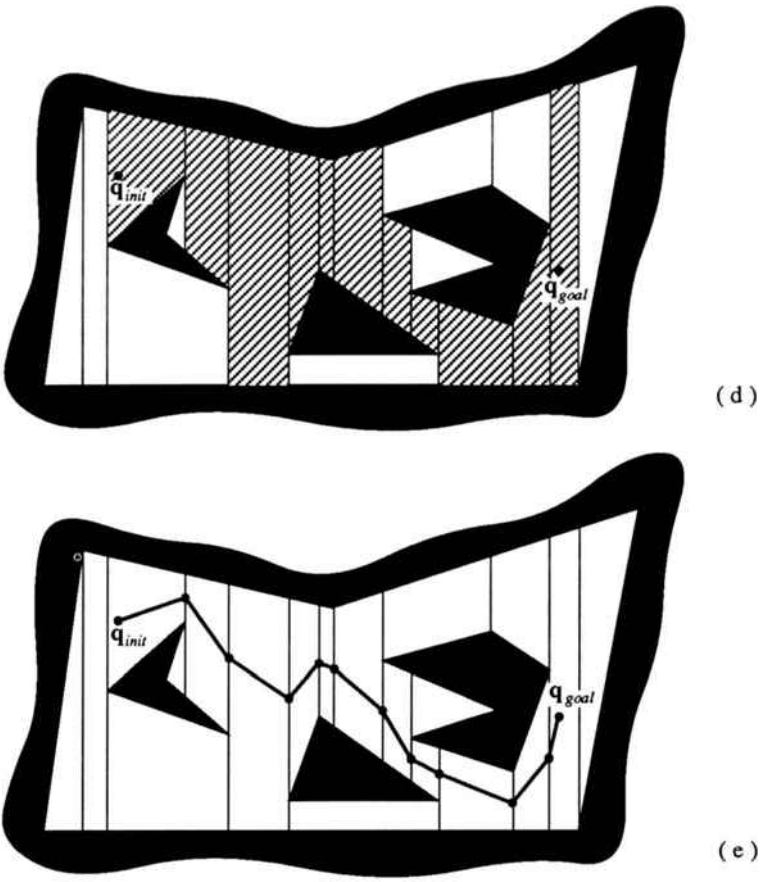


Figure 6.2. A path in the connectivity graph determines a channel (the sequence of striped cells in Figure d) in free space. The third step of the method consists of transforming this channel into a free path, for example by connecting the initial configuration to the goal configuration through the midpoints of the intersections of every two successive cells (Figure e).

cells whose interiors lie entirely in the free space are used to construct the connectivity graph. If the search of this graph terminates successfully, a path in the free space is easily generated. Otherwise, it may mean that either the resolution of the decomposition is insufficient, or no free path exists between the initial and goal configurations. Often, an approximate cell decomposition method operates in a hierarchical fashion by using a

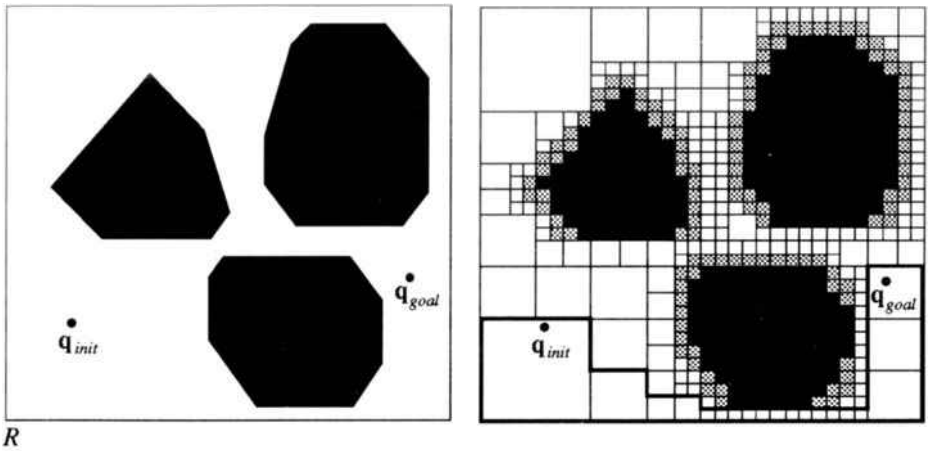


Figure 7. This figure illustrates an approximate cell decomposition method. The free space is externally bounded by a rectangle R and internally bounded by three polygons (Figure a). The rectangle R is decomposed into four identical rectangles. If the interior of a rectangle lies completely in free space or in the C-obstacle region, it is not decomposed further. Otherwise, it is recursively decomposed into four rectangles until some predefined resolution is attained. The result is shown in Figure b. White cells lie outside the C-obstacle region; black cells lie inside the C-obstacle region; grey cells are cut by the C-obstacle region. This kind of decomposition is called a "quadtree" decomposition. A channel extracted from this decomposition is shown in bold contour.

rather coarse resolution at the beginning, and refining it until either a free path is found or a limit resolution is attained.

Provided that they are equipped with both appropriate search techniques and exact numerical computation techniques, exact cell decomposition methods are complete — i.e. they are guaranteed to find a free path whenever one exists and to return failure otherwise. Approximate methods may not be complete; but, for most of them, the precision of the approximation can be tuned and made arbitrarily small (at the expense of the running time), so that the methods are said to be "resolution-complete". On the other hand, except in very simple cases, exact methods are mathematically more involved than approximate ones. Hence, the latter are usually easier to implement.

TESLA EXHIBIT 1013, Page 18

4.3 Potential Field

A straightforward approach to motion planning is to discretize the configuration space into a fine regular grid of configurations and to search this grid for a free path. This approach requires powerful heuristics to guide the search, since the grid is in general enormous. Several types of heuristics have been proposed. The most successful ones take the form of functions that are interpreted as *potential fields.*

The metaphor suggested by this terminology is that the robot represented as a point in configuration space is a particle moving under the influence of an artificial potential produced by the goal configuration and the C-obstacles. Typically the goal configuration generates an "attractive potential" which pulls the robot toward the goal, and the C-obstacles produce a "repulsive potential" which pushes the robot away from them. The negated gradient of the total potential is treated as an artificial force applied to the robot. At every configuration, the direction of this force is considered the most promising direction of motion.

Figure 8 illustrates the notion of attractive and repulsive potentials, and their combination. The attractive potential (Figure 8.b) is a quadratic well with its minimum at the goal configuration. The repulsive potential (Figure 8.c) is non-zero only within some distance from the C-obstacles and tends toward infinity when the distance to the C-obstacles tends toward zero (hence, is truncated in the figure). A path between the initial and goal configurations is constructed by tracking the negated gradient of the total potential (Figure 8.e). A matrix of orientations of the negated gradient vector field (orientations of the artifical forces induced by the potential field) is displayed in Figure 8.f.

In comparison to other methods, potential field methods can be very efficient. However, they have a major drawback. Since they are essentially fastest descent optimization methods, they can get trapped into local minima of the potential function other than the goal configuration. One attack of this problem is to design potential functions that have no local minima other than the goal configuration in the connected subset of free space which contains the goal configuration. Another approach is to complement the basic potential field approach with powerful mechanisms to escape from local minima.

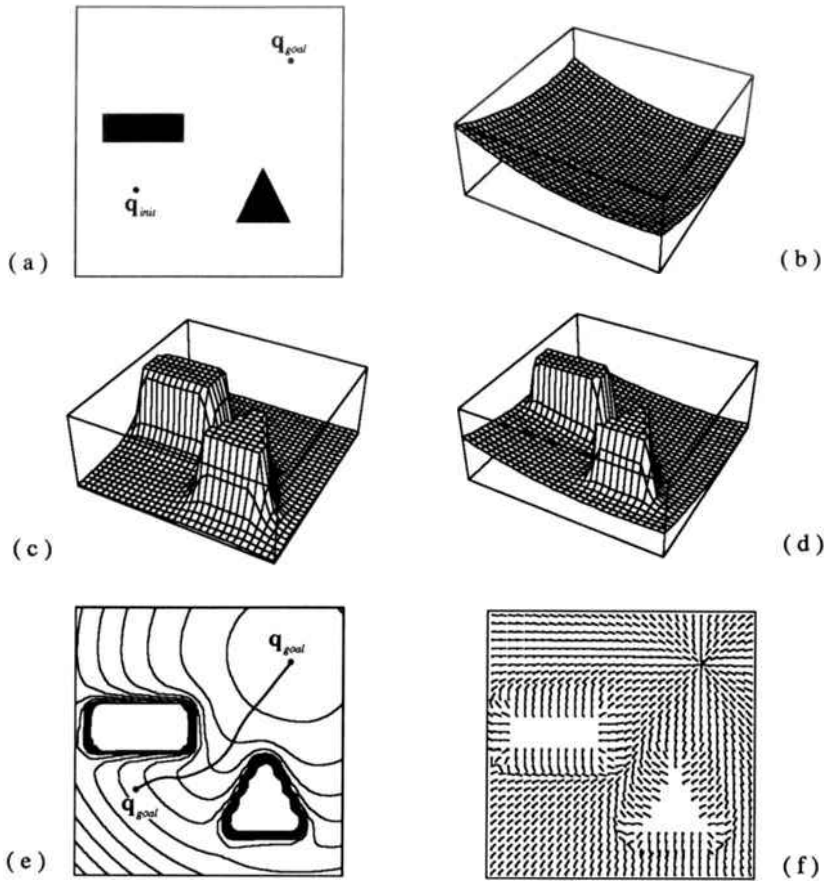


Figure 8. This series of diagrams illustrates the potential field approach. A simple two-dimensional configuration space with two polygonal C-obstacles is depicted in Figure a. Figure b shows an attractive potential generated by the goal configuration $\mathbf{q}_{goal}$. Figure c shows a repulsive potential generated by the C-obstacles. Figure d shows the sum of the two potentials. Figure e displays equipotential contours of the total potential and a path obtained by following its negated gradient. Figure f shows orientations of the negated gradient of the total potential over free space.

4.4 Global versus Local Methods

The roadmap and cell decomposition methods reduce the problem of finding a continuous free path to that of searching a graph (e.g. the

visibility graph, the Voronoi diagram, the connectivity graph) by first analyzing the connectivity of the free space. Unlike these methods, potential field methods do not include an initial processing step aimed at capturing the connectivity of the free space in a concise representation. Instead, they search a much larger graph representing the adjacency among the configurations contained in the grid thrown over configuration space. At each step, they move from one configuration in the grid to another one, basing their choice on the computation of the potential gradient. It is typical for this gradient to depend only on the contents of the configuration space in the neighborhood of the current configuration of the robot. (As illustrated in Figure 8.c, the repulsive potential is usually defined with a limited range of influence around the C-obstacles.) Therefore, potential field methods are often called *local* methods, while the roadmap and cell decomposition methods are called *global* methods.

In practice, this distinction is certainly valid. For example, a potential field method can be run while the robot is moving, by tracking the negated gradient of a potential field defined as a function of both the distance to the goal configuration and the distances to the obstacles as measured by range sensors. The distinction is nevertheless intuitive and has no solid theoretical basis. A potential field method may use a potential function that is free of local minima (in the connected subset containing the goal configuration). Computing such a function certainly requires the geometry of the whole free space to be taken into account. Then the method is as global as any roadmap or cell decomposition method. On the other hand, a roadmap or cell decomposition method can be made local by restricting its application to a subset of the configuration space around the current configuration of the robot. A path is then generated in an iterative fashion by concatenating subpaths, each achieving an intermediate configuration in such a restricted subset.

5 Extensions of the Basic Problem

The basic motion planning problem makes assumptions that significantly limit the practicality of its solutions. In other words, it may be quite difficult to reduce an actual robotic problem to an instance of the basic problem, to solve this instance, and to adapt the produced solution in order to match the conditions of the original problem. In this section we

introduce and discuss informally several important extensions that we will investigate in detail in Chapters 8 through 11. Other extensions can certainly be envisioned, but they have received too little attention so far to be treated in this book.

5.1 Multiple Moving Objects

In the basic problem we assumed that the obstacles were fixed, that there was a single robot, and that this robot was made of a single rigid object. Under the title "multiple moving objects", we consider a series of extensions that removes these assumptions. One extension consists of including *moving obstacles* in the workspace. The second one allows *multiple robots* to operate in the same workspace, each being a potential obstacle to the others. The third extension considers *articulated robots*, which are made of several rigid objects connected by joints.

The first extension (and possibly the second) requires time to be explicitly considered in the motion plans. The second and third extensions yield configuration spaces of arbitrarily large dimensions.[3]

Moving Obstacle. In the presence of moving obstacles, the motion planning problem can no longer be solved by merely constructing a geometric path. A continuous function of time specifying the robot's configuration at each instant must be generated instead. This can be done by adding one dimension representing time to the robot's configuration space. The new space, denoted by $\mathcal{CT}$, is called *configuration-time space*[4]. The workspace obstacles map in $\mathcal{CT}$ to static regions called *CT-obstacles*. Every cross-section through $\mathcal{CT}$ at time t is the configuration space of the robot at instant t. It cuts the CT-obstacles according to the C-obstacles corresponding to the workspace obstacles at their locations at instant t. Motion planning consists of finding a path among the CT-obstacles in $\mathcal{CT}$. Since time is irreversible, this path must have the property of never going backward along the time axis. Hence, path planning methods have to be modified in order to take this specificity of the time dimension into account.

[3] In Section 6 we will see that the inherent computational complexity of path planning is exponential in the dimension of the robot's configuration space.

[4] In the literature it is also called configuration space-time.

TESLA EXHIBIT 1013, Page 22

If there is no constraint on the robot's velocity and acceleration, and if the motion of every obstacle is known beforehand, it is rather straightforward to extend some basic planning methods to handle this new problem. If constraints apply to the robot's velocity and/or its acceleration, they translate to geometric constraints on the slope and the curvature of a path along the time dimension, and the problem turns out to be significantly harder.

Multiple Robots. Motion planning with multiple robots differs from planning in the presence of moving obstacles in that the motions of the robots have to be planned, while the motions of the moving obstacles are not under control.

One way to deal with multiple robots $\mathcal{A}_1, ..., \mathcal{A}_p$ operating in the same workspace $\mathcal{W}$ is to treat them as a single multi-bodied robot $\mathcal{A} = \{\mathcal{A}_1, ..., \mathcal{A}_p\}$. The *composite configuration space* of $\mathcal{A}$ is $\mathcal{C} = \mathcal{C}_1 \times ... \times \mathcal{C}_p$, i.e. the set product of the configuration spaces $\mathcal{C}_i$ of the individual robots $\mathcal{A}_1, ..., \mathcal{A}_p$. Every configuration in $\mathcal{C}$ determines a unique position and orientation for each robot $\mathcal{A}_i$. C-obstacles in this space result from the interaction of either a robot $\mathcal{A}_i$ and an obstacle $\mathcal{B}_j$, or two robots $\mathcal{A}_i$ and $\mathcal{A}_j$. Basic path planning methods can be used to plan a path of $\mathcal{A}$ in $\mathcal{C}$. This approach to multi-robot path planning is called *centralized planning*. A difficulty is that it may yield high-dimensional configuration spaces. Indeed, the dimension of the composite configuration space $\mathcal{C}$ is equal to the sum of the dimensions of the individual configuration spaces $\mathcal{C}_1$ through $\mathcal{C}_p$.

Another approach to motion planning with multiple robots, *decoupled planning*, is to plan the motion of each robot more or less independently of the other robots and to consider the interactions among the paths in a second phase of planning. By doing so, the amount of computation may be significantly reduced, but at the expense of completeness. Figure 9 illustrates a planning problem where the decoupled approach is likely to fail. The problem consists of interchanging the positions of two robots (discs) in a narrow corridor where they cannot pass each other. There is enough room at one end of the corridor for a permutation of the two robots. However, by considering each robot separately, the decoupled planning approach has no mechanism to infer that both robots must first move to this end of the corridor.

TESLA EXHIBIT 1013, Page 23

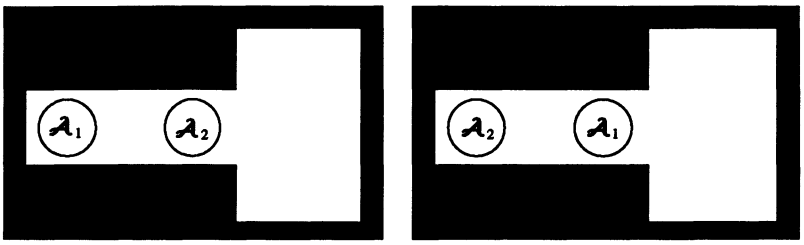


Figure 9. Two discs $\mathcal{A}_1$ and $\mathcal{A}_2$ have to interchange their positions in a narrow corridor where they cannot pass each other. However, there is enough room for permutation at one end of the corridor. Decoupled planning, which considers the two robots separately, would very likely fail to solve this problem.

Articulated Robots. An articulated robot $\mathcal{A}$ is made of several moving rigid objects $\mathcal{A}_1, ..., \mathcal{A}_p$ (called *links*) connected by *joints*, e.g. revolute joints (hinges) and prismatic joints (sliding joints). Each joint constrains the relative movements of the two objects it connects. A typical example of an articulated robot is a robot arm (see Figure 10).

$\mathcal{A}$ can be seen as a set of p moving rigid objects. The constraints imposed by the joints on the relative movements of the $\mathcal{A}_i$'s determine a subset of the composite configuration space of these objects, which is the actual configuration space of $\mathcal{A}$. Every configuration in this subset indeed determines a unique position and orientation for each of the p links. This subset is in general relatively easy to parameterize, for example by associating a distance or an angle with each joint. Several basic motion planning methods extend to this case in a straightforward fashion, at least in theory. A practical problem that has to be faced is that the dimension of the configuration space grows with the number of joints.

Figure 11 illustrates several configurations of a planar articulated robot along a path generated by a potential field method. In this example, the robot is a sequence of eight links. All the joints are revolute joints. Notice that for an articulated robot with many links and joints, it may be problematic to specify a free goal configuration. In the example of Figure 11, only the goal position of the endpoint of the arm was specified to the planner. This corresponds to defining a goal region in the robot's configuration space, rather than a unique goal configuration.

TESLA EXHIBIT 1013, Page 24

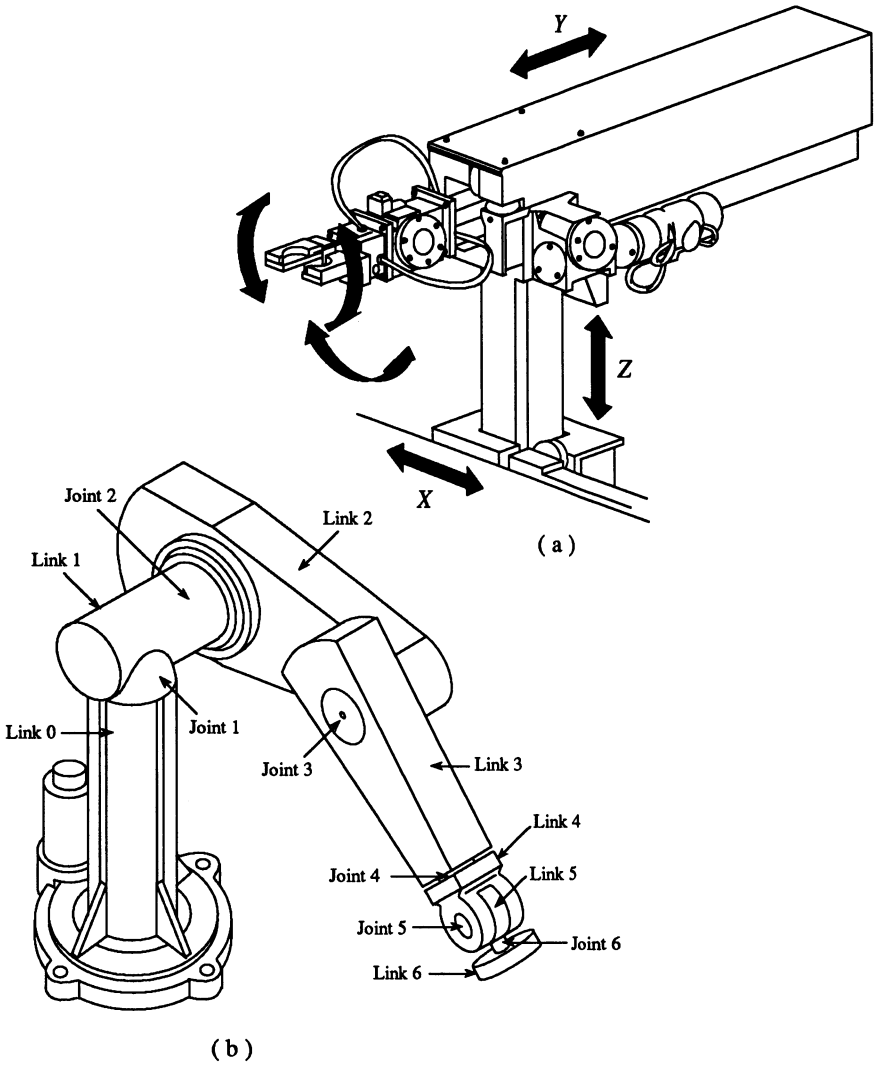


Figure 10. A robot arm is a typical example of an articulated robot. It is made of several rigid objects, called links, serially connected by joints. Most robot arms have revolute joints and/or prismatic joints. A revolute joint constrains the relative motion of two objects to be a rotation around an axis fixed with respect to both objects. A prismatic joint constrains the relative motion of two objects to be a translation along an axis fixed with respect to both objects. Figure a shows a Cartesian robot, with three prismatic joints and three revolute ones. Figure b shows a robot with six revolute joints.

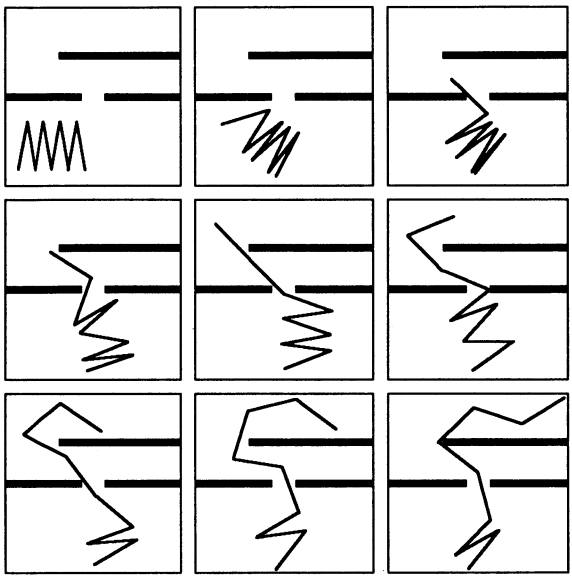


Figure 11. This figure shows a sequence of configurations of a planar eight-revolute-joint arm along a path generated using a potential field method.

5.2 Kinematic Constraints

In the basic problem we assumed that the robot was a free-flying object, i.e. the only constraints on its motions were due to the obstacles. In some problems we may want to impose additional kinematic constraints to the robot's motions. Two types of constraints, called *holonomic constraints* and *nonholonomic constraints*, can be considered.

Holonomic Constraints. Let us assume that a configuration is represented by a list of parameters of minimal cardinality (see Section 3). A holonomic equality constraint is an equality relation among these parameters, which can be solved for one of the parameters. Such a relation reduces the dimension of the actual configuration space of the robot by one. A set of k independent holonomic constraints reduces it by k.

Consider for example the case of a three-dimensional object $\mathcal{A}$ allowed to translate freely, but constrained to rotate around a fixed axis relative to $\mathcal{A}$. Let us represent the orientation of $\mathcal{A}$ by three angles. The

constraints restricting the possible orientations of $\mathcal{A}$ can be expressed as two independent equations among these angles. While the dimension of the configuration space of a free-flying object in a three-dimensional workspace is 6, $\mathcal{A}$'s configuration space has dimension 4. The particular case where $\mathcal{A}$ can translate freely at fixed orientation could also be considered as a problem with holonomic constraints. However, since this problem is equivalent to the problem of planning the motion of a point robot in $\mathbf{R}^N$, we regard it as a particular case of the basic motion planning problem.

Articulated robots provide another example of holonomic constraints. For instance, a revolute joint determines two holonomic constraints. Indeed, while six parameters are necessary to define the configuration of two free-flying planar objects, four parameters suffice to determine the positions and orientations of two planar objects connected by a revolute joint. As another example, consider a six-joint arm carrying a glass of water. Requiring the glass to stay vertical during the motion corresponds to imposing two holonomic constraints to the arm.

Holonomic constraints certainly affect the definition of the robot's configuration space and may even change its global connectedness. Nonetheless, holonomic constraints do not raise new fundamental issues. Many basic planning methods remain applicable.

Nonholonomic Constraints. A nonholonomic equality constraint is a non-integrable equation involving the configuration parameters and their derivatives (velocity parameters). Such a constraint does not reduce the dimension of the space of configurations attainable by the robot, but reduces the dimension of the space of possible differential motions (i.e. the space of the velocity directions) at any given configuration.

Consider for example a car-like robot $\mathcal{A}$ rolling on a flat ground. We model it as a rectangular object moving in $\mathcal{W} = \mathbf{R}^2$ (see Figure 12). We know by experience that in an empty space we can drive the robot to any position with any orientation. Hence the robot's configuration space has three dimensions, two of translation and one of rotation. Let us represent a configuration of $\mathcal{A}$ by (x, y, θ), where x and y are the coordinates, in the frame $\mathcal{F}_\mathcal{W}$, of the midpoint R between the two rear wheels and $\theta \in [0, 2\pi)$ is the angle between the x-axis of $\mathcal{F}_\mathcal{W}$ and the main axis of $\mathcal{A}$. At any instant during a motion, assuming no slipping,

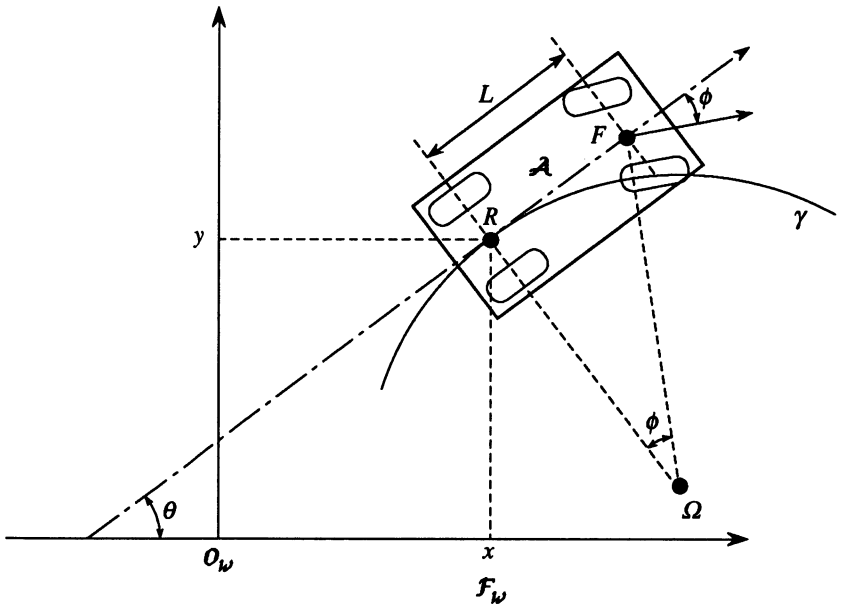


Figure 12. This car-like robot $\mathcal{A}$ is modeled as a rectangle. Any configuration of $\mathcal{A}$ is represented by three parameters $(x, y, \theta) \in \mathbf{R}^2 \times [0, 2\pi)$, with modulo 2π arithmetic on θ. Although in the absence of obstacles the robot can attain any configuration in $\mathbf{R}^2 \times [0, 2\pi)$, the projection of any path of $\mathcal{A}$ on the xy-plane is a curve γ tangent to the main axis of $\mathcal{A}$, thus implying that the relation $-\sin\theta\, dx + \cos\theta\, dy = 0$ be verified. This relation is a nonholonomic equality constraint. The curvature of γ is $L/\tan\phi$, where L is the distance between R and F, the midpoint between the two front wheels, and ϕ is the steering angle, i.e. the angle between $\mathcal{A}$'s main axis and the velocity vector of point F. In general, $|\phi| \leq \phi_{max} < \pi/2$.

the velocity of R has to point along the main axis of $\mathcal{A}$. Therefore, the motion is constrained by the relation:

$$-\sin\theta\, dx + \cos\theta\, dy = 0.$$

It can be shown that this equation is non-integrable, hence is a nonholonomic equality constraint. Due to this constraint, the space of differential motions $(dx, dy, d\theta)$ of the robot at any configuration (x, y, θ) is a two-dimensional space. If the robot was a free-flying object, this space would be three-dimensional. The instantaneous motion of the car-like robot is

TESLA EXHIBIT 1013, Page 28

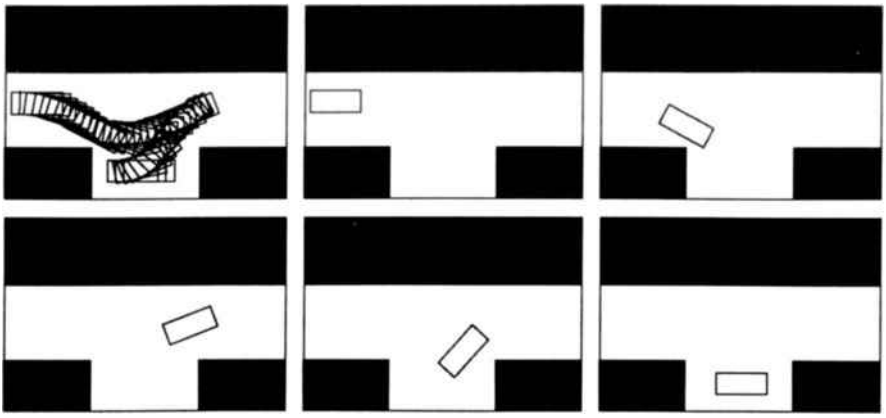


Figure 13. This figure illustrates a path generated by a planner described in Chapter 9 for a car-like robot in order to perform a parallel parking operation.

determined by two parameters: the linear velocity along its main axis and the steering angle. However, when the steering angle is non-zero, the robot changes orientation, and its linear velocity with it, allowing the robot's configurations to span a three-dimensional space.

Furthermore, the steering angle denoted by ϕ in Figure 12 is in general constrained to take values in an interval $[-\phi_{max}, +\phi_{max}]$, with $\phi_{max} < \pi/2$. This constraint can be rewritten as an inequality involving dx, dy and $d\theta$. It restricts the set of possible differential motions without changing its dimension. We call it a nonholonomic inequality constraint.

Nonholonomic constraints restrict the geometry of the feasible free paths between two configurations. They are much harder to deal with in a planner than holonomic constraints. Figure 13 illustrates a path generated for a car-like robot in order to perform a parallel parking operation.

5.3 Uncertainty

The basic problem assumes that the robot can follow the paths generated by the planner exactly. It also assumes that the geometry of the robot, the geometry of the obstacles, and the location of the obstacles are accurately known. No robot setting rigorously satisfies these assumptions, and both robot control and geometric models are imperfect. Fortunately,

we often do not have to worry about these imperfections because they are small relative to the tolerance of the task being performed. But this is not always the case.

At the other extreme, the robot could have little or no prior knowledge about its workspace. Then, it would have to rely on its sensors at execution time to obtain the information needed to accomplish the task. This extreme case requires the robot to explore its workspace, and it tends to be outside the realm of motion planning. The robot may interweave planning and execution monitoring activities, but the more incomplete the prior knowledge, the less important the role of planning.

An intermediate situation is when there are errors in robot control and in the initial geometric models, but these errors are contained within bounded regions. For example, the actual obstacles' locations are slightly different from those in the robot's model, but the errors on the parameters defining these locations are bounded. Similarly, the robot moves along a direction that is different from the commanded one, but the actual direction of motion is contained in a narrow cone centered along the commanded direction. In order to deal with such bounded errors, we assume that the robot is equipped with sensors that can be used at execution time to acquire additional information. But sensors are not perfect, either. For example, a position sensor does not return the exact configuration of the robot. Again, it is realistic to assume that the sensing errors are contained in bounded uncertainty regions. When errors in control, sensing and model are reasonably small, it is interesting to generate motion plans that are tolerant to these errors, i.e. that achieve goals reliably by anticipating the various possible contingencies.

The motion planning problem with bounded uncertainty can be stated as follows: Given an *initial region* $\mathcal{I}$ and a *goal region* $\mathcal{G}$ in the robot's configuration space, generate a motion plan whose execution guarantees the robot to reach a configuration in $\mathcal{G}$ if it starts from any (unknown) configuration in $\mathcal{I}$, despite bounded uncertainty in control, sensing and model. A solution to this problem is a plan that combines motion commands and sensor readings that interact at execution time in order to reduce uncertainty and guide the robot toward the goal.

Planning in the presence of bounded uncertainty raises new issues explored neither in the basic problem, nor in the previous extensions. Due

TESLA EXHIBIT 1013, Page 30

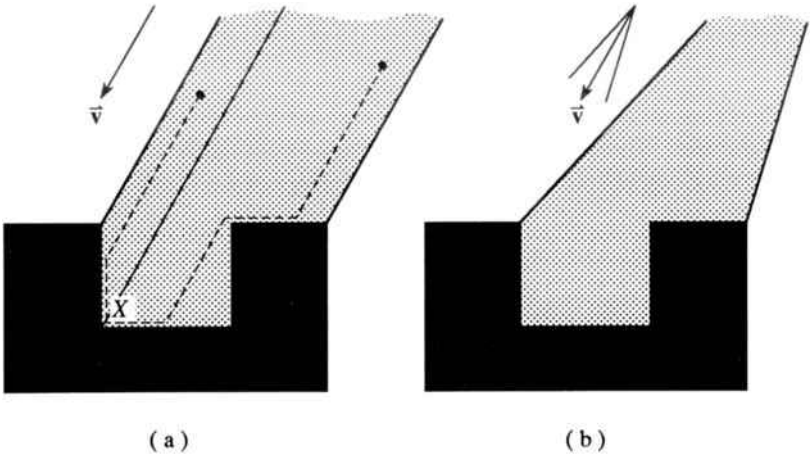


Figure 14. This figure illustrates a force-compliant motion in a two-dimensional configuration space with a polygonal C-obstacle. The robot is commanded to move along the direction $\vec{v}$. Assuming perfect control (Figure a), the robot moves along a straight line parallel to $\vec{v}$ as long as it is in free space. When it touches an obstacle, the robot switches to a path following the projection of $\vec{v}$ in the C-obstacle's boundary. The grey region in Figure a shows the range of initial configurations from which any compliant motion commanded along $\vec{v}$ reaches the vertex X. The two dashed lines illustrate two such motions. (The range of initial configurations from which a pure position-controlled motion commanded along $\vec{v}$ reaches X is the half-line parallel to $\vec{v}$ ending at X). If control is imperfect, e.g. the robot moves along a direction contained in a cone centered along the commanded direction $\vec{v}$ (see Figure b), the grey region is slightly reduced. (In contrast, the range of initial configurations from which a pure position-controlled motion commanded along $\vec{v}$ is guaranteed to reach X vanishes to the point X itself.)

to uncertainty in control, a motion command may produce any path among the infinitely many ones which are consistent with both the command and the uncertainty. In order for the planner to guarantee that execution will succeed, all of these paths must reach the goal (goal reachability). But this is only a sufficient condition. The plan must also terminate *in* the goal. Due to uncertainty in sensing, it may not be a trivial issue to recognize goal achievement reliably. The planner must make sure that enough information will be made available to the controller

during execution (goal recognizability).

Uncertainty often leads to using sensor-based motion commands whose behavior is less sensitive to errors than purely position-controlled motion commands. *Force-compliant* motion commands are one example of such commands. When they are used, the robot may touch obstacle surfaces and slide on them (along the projection of the commanded direction of motion), rather than just stop. Figure 14 illustrates a compliant motion of a robot commanded along the direction $\vec{v}$ in a two-dimensional configuration space. In Figure 14.a we assume perfect control; in Figure 14.b we assume imperfect control (the actual direction of motion is contained in a cone centered along $\vec{v}$). In both cases, the grey region is the set of initial configurations from where the robot commanded along $\vec{v}$ is guaranteed to reach the C-obstacle vertex X (assuming frictionless edges). The grey region in Figure 14.b is only slightly smaller than in Figure 14.a. In contrast, with uncertainty in control, no position-controlled motion commanded along $\vec{v}$ is guaranteed to reach X. Planning such sensory-based motion commands may require the physics of the workspace to be taken into consideration. For example, when force sensing is used to detect contact with the obstacles and slide in their boundary, frictional effects ignored in Figure 14 may be critical.

5.4 Movable Objects

In the basic problem the objects in the workspace other than the robot are all stationary, non-movable obstacles. In one extension introduced in Subsection 5.1 the obstacles can move, but their motions are not under the robot's control. Another important extension is when the workspace contains *movable objects*, i.e. objects that the robot can displace, for example by grasping them.

The presence of movable objects requires the generation of sequences of motion commands, each intended to achieve some intermediate goal which may not have been specified in the input problem. For example, even if there exists no feasible path to a goal configuration in some given arrangement of the workspace, the robot may create one by moving movable objects to other locations. In order to do so, the planner must select these other locations (intermediate goals). Hence, a motion plan appears as an alternate sequence of *transit* and *transfer* motions. Along a transit motion the robot moves alone. Along a transfer motion it

TESLA EXHIBIT 1013, Page 32

moves together with a movable object. The transit and transfer motions lie in different spaces. The transit motions occur in the configuration space of the robot, but different arrangements of the movable objects yield different geometries in this space. A transfer motion occurs in the configuration space of the union of the robot and the movable object moving with it. The main problem for the planner is to determine which spaces to use and when to change from one space to another.

The most common way for a robot to displace a movable object is to rigidly grasp it and to move with it. For example, if the robot is equipped with a multi-fingered gripper, grasping an object consists of placing the fingers of the gripper on the surface bounding the object and exerting some gripping force. Then, if the robot moves, the object moves with it as a single object. Grasping requires that the robot be capable of selecting the positions of the fingers on the object (another intermediate goal). Several constraints — force-closure, stability, tolerance to positioning errors, minimization of gripping forces, geometric feasibility — have to be satisfied by these positions. The grasp planning problem (a difficult problem in its own) is part of the motion planning problem in the presence of movable objects.

Grasping is not the only way of moving a movable object. The robot may also push an object on a table. It may also cause an object to move by tilting the surface supporting the object with respect to gravity. If it is equipped with a dexterous hand (i.e. a gripper with independent multi-joint fingers), it may also manipulate the object within its hand. However, these operations are very difficult to model and existing models still lack the required precision for constructing effective planners.

Movable objects allow new types of planning problems to be considered, where the important goals are not just to achieve given configurations of the robot. Arrangements of movable objects, such as assembling parts into a composite object or arranging pieces of furniture according to a given pattern, can now be the real goals.

6 Computational Complexity

Instances of the same type of motion planning problem may considerably differ in their "size", e.g. the dimension of configuration space and the

number of obstacles in the workspace. It is important to quantify the performance of the planning methods — typically, the time they require to solve a problem — and the inherent computational complexity of the planning problems as a function of these parameters. Analysis of the methods is useful to assess their practicality and to detect opportunities for increased efficiency. Analysis of the problems is useful to suggest new ways of formulating them when there is substantial evidence that the original formulation only admits too costly solutions.

In this section we review important theoretical results relative to the computational complexity of robot motion planning. We mainly consider lower bounds on the time complexity of problems. In the following chapters we will analyze the complexity of specific planning methods (upper bounds). Basic notions in computational complexity analysis are surveyed in Appendix B.

The first lower bound was established by Reif for planning free paths in a configuration space of arbitrary dimension ("generalized mover's problem") [Reif, 1979]:

Planning a free path for a robot made of an arbitrary number of polyhedral bodies connected together at some joint vertices, among a finite set of polyhedral obstacles, between any two given configurations, is a PSPACE-hard problem.

PSPACE-hardness has also been established for a variety of path planning problems with simpler or more specific robotic systems, for example:

- *Planar linkages.* A planar linkage consists of an arbitrary number of one-dimensional rigid links connected by revolute joints, each joint connecting two or more links. The locations of some joints are fixed. Although all the links move in the same plane, they are allowed to cross each other. In the absence of obstacles, deciding whether a planar linkage in some initial configuration can be moved so that a certain joint reaches a given point in the plane is PSPACE-hard [Hopcroft, Joseph, and Whitesides, 1984]. This result reflects the complexity of the connectivity of the configuration space of a planar linkage.

- *Multiple rectangles.* The robotic system consists of arbitrarily many rectangles in an empty rectangular workspace. Each rectangle can only move in translation with its sides parallel to the sides of the

workspace boundary. The problem is to plan the coordinated motion of the rectangles between two given configurations, so that they do not intersect. This problem, which is equivalent to planning a path in the configuration space of the multi-bodied robot consisting of all the rectangles, is both PSPACE-hard [Hopcroft, Schwartz and Sharir, 1984] and in PSPACE [Hopcroft and Wilfong, 1986b]. Hence, it is PSPACE-complete.

- *Planar arm.* Consider a planar arm consisting of arbitrarily many links serially connected by revolute joints such that all the links are constrained to move in the plane. Planning a free path for such an arm among a finite number of polygonal obstacles, between any two given configurations, is PSPACE-hard [Joseph and Plantiga, 1985].

The following result provides an upper bound on path planning complexity:

A free path in a configuration space of any fixed dimension m, when the free space is a set defined by n polynomial constraints of maximal degree d, can be computed by an algorithm whose time complexity is exponential in m and polynomial in both n ("geometrical complexity") and d ("algebraic complexity").

This bound was first established by Schwartz and Sharir [Schwartz and Sharir, 1983b], who gave an exact cell decomposition method based on the Collins decomposition algorithm. This method, which we will present in Chapter 5, is twice exponential in m. The most efficient algorithm to date is due to Canny [Canny, 1988] who proposed a roadmap method that is singly-exponential in m. Both methods reduce the planning problem to the problem of deciding the satisfiability of sentences in the first-order theory of the reals. As we will see in several chapters, the theory of the reals has been a powerful tool in establishing upper bounds for motion planning problems. Nonetheless, there exist no implementations of the decision algorithms that are fast enough to constitute practical solutions to these problems.

The above results strongly suggest that the complexity of path planning increases exponentially with the dimension of the configuration space. Although this statement is likely to be true for most robotic systems, there nevertheless exist simple settings for which it does not hold. For example, a polynomial time algorithm has been proposed to plan the

paths of a planar arm with arbitrarily many joints within a circle and without obstacles other than the circle [Hopcroft, Joseph and Whitesides, 1985] [Kantabutra and Kosaraju, 1986].

On the other hand, if the dimension of the configuration space is relatively small and/or we accept incomplete planning algorithms (i.e. algorithms that may fail to generate a path, even if one exists), specific algorithms of reasonable complexity can be designed. For example, if the dimension of configuration space is limited to two, it has been shown that, under reasonable assumptions, the basic motion planning problem can be solved in time $O(\lambda_s(n)\log^2 n)$, where n is the number of polynomial constraints defining the free space, s is a parameter depending in particular on the maximum degree of these constraints, and $\lambda_s(n)$ is an almost linear function of n (the length of (n, s) Davenport-Schinzel sequences) [Guibas, Sharir and Sifrony, 1988].

Interesting results have also been obtained with moving obstacles:

Planning the motion of a rigid object translating without rotation in three dimensions among arbitrarily many moving obstacles that may both translate and rotate is a PSPACE-hard problem if the velocity modulus of the object is bounded, and an NP-hard problem otherwise.

This result was established in [Reif and Sharir, 1985]. A simpler problem — planning the motion of a point in the plane, with bounded velocity modulus, among arbitrarily many convex polygonal obstacles moving at constant linear velocity without rotation — has also been shown to be NP-hard [Canny, 1988]. However, if the velocity of the robot is not constrained and if the motions of the obstacles can be described algebraically, the problem admits a polynomial time algorithm.

Concerning motion planning with uncertainty, the following result has been established by Canny and Reif [Canny and Reif, 1987]:

Planning compliant motions for a point in the presence of uncertainty, in a three-dimensional polyhedral configuration space with an arbitrarily large number of faces, is a nondeterministic exponential time hard (NEXPTIME-hard) problem.

This result means that the problem is expected to require at least exponential time on a nondeterministic machine, hence doubly-exponential time on a deterministic machine. Previously, Natarajan [Natarajan,

1986] had shown that this problem is PSPACE-hard. With a few assumptions allowing the reduction of the planning problem in an m-dimensional configuration space to a decision problem in the theory of the reals, an r-step plan (if any) can be generated by an algorithm of complexity $2^{2^{O(rnm)}}$, where n is the number of polynomial constraints defining the boundary of the C-obstacle region [Canny, 1989]. More tractable subclasses of motion planning in the plane in the presence of uncertainty have been identified.

7 Reduction of Complexity

The complexity results surveyed in the previous section give strong evidence that the time required to solve a motion planning problem increases quickly with the dimension of the configuration space, the number of polynomial constraints on the robot's motion, and the degree of these constraints. Therefore, it is important that these parameters be as small as possible.

To that purpose, one may approximate the real problem by a simplified, but still realistic problem, before submitting it to the motion planner. If the robot task is reasonably well understood and if it is acceptable to trade some generality against better time performance, such a simplification is almost always feasible. The simplified problems can be provided by the user, or automatically generated by the planner from the original input description.

The following subsections present techniques for approximating motion planning problems into simpler ones. Most of the discussion focuses on planning free paths for rigid objects and robot arms. However, some of the techniques can also be applied to other motion planning problems.

7.1 Projection in Configuration Space

One can reduce the dimension of the configuration space $\mathcal{C}$ by replacing the robot $\mathcal{A}$ by the surface or volume it sweeps out when it moves along r independent axes. This corresponds to projecting the m-dimensional configuration space $\mathcal{C}$ along r of its dimensions. Let $\mathcal{C}'$ be the projected space. Its dimension is $m - r$. The C-obstacles in $\mathcal{C}$ are projected into $\mathcal{C}'$, and path planning is conducted in $\mathcal{C}'_{free}$, the free space in $\mathcal{C}'$.

Examples: - Consider a two-dimensional object $\mathcal{A}$ that can both translate and rotate freely in the plane. $\mathcal{A}$ can be replaced by a disc $\mathcal{A}'$ containing $\mathcal{A}$ (ideally, this disc is bounded by the minimum spanning circle of $\mathcal{A}$, i.e. the smallest circle that encloses all the points of $\mathcal{A}$). This approximation reduces the configuration space dimension from 3 to 2. It is a realistic one only when $\mathcal{A}$ is not an elongated object.

- Consider a robot arm with six revolute joints, the axes of the last three joints intersecting at a single point. It is rather typical to replace the last three links, the end-effector and the payload by the volume they sweep out when the last three revolute joints span their angular ranges. This reduces the configuration space dimension from 6 to 3. This approximation is realistic only when the payload is not a long object. ■

If a free path τ is found in $\mathcal{C}'$, it determines a family of free paths in $\mathcal{C}$. The configuration parameters not specified in τ can be selected arbitrarily. But, if no free path is found in $\mathcal{C}'$, this does not guarantee that there is no free path in $\mathcal{C}$.

If most of the motions of $\mathcal{A}$ occur in a rather uncluttered workspace and if the ratio of the area/volume of $\mathcal{A}$ by the area/volume of $\mathcal{A}'$ is close to 1, it is probably more efficient to plan a motion for $\mathcal{A}'$, rather than for $\mathcal{A}$. It is also possible to build a two-stage planner that plans the path of $\mathcal{A}$ only when it previously failed to plan the path of $\mathcal{A}'$.

7.2 Slicing in Configuration Space

Another way to reduce the dimension of configuration space is to consider a cross-section of dimension $m - r$ through $\mathcal{C}$ and to plan a free path in the space defined by this cross-section. This corresponds to forbidding the motion of $\mathcal{A}$ along r axes.

Examples: - Let $\mathcal{A}$ be a free-flying three-dimensional object. In order to plan a free path from the initial configuration $\mathbf{q}_{init} = (\mathcal{T}_{init}, \Theta_{init})$ to the goal configuration $\mathbf{q}_{goal} = (\mathcal{T}_{goal}, \Theta_{goal})$, one may consider the intermediate configuration $\mathbf{q}_{inter} = (\mathcal{T}_{goal}, \Theta_{init})$ at which $\mathcal{A}$ has the same orientation Θ_{init} as at the initial configuration and the same position $\mathcal{T}_{goal}$ as at the goal configuration. Then, the planner may attempt to generate a first path from $\mathbf{q}_{init}$ to $\mathbf{q}_{inter}$ with a fixed orientation of $\mathcal{A}$ and a second path from $\mathbf{q}_{inter}$ to $\mathbf{q}_{goal}$ with a fixed position of $\mathcal{A}$. Each of the two paths lies in a three-dimensional cross-section of the configuration

TESLA EXHIBIT 1013, Page 38

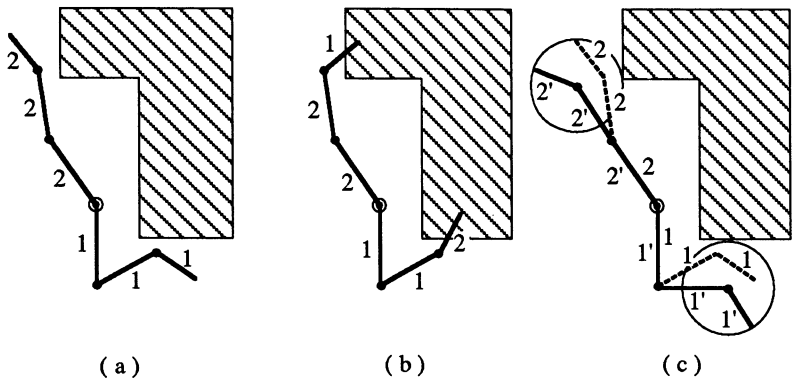


Figure 15. A planar three-joint arm has to move from the initial configuration marked **1** to the goal configuration marked **2**. Both configurations are shown in Figure a. The double-circled joint is fixed and there is a single obstacle. When the first two links are in the initial (resp. goal) configuration, and the third link in the goal (resp. initial) configuration, the arm collides with the obstacle as shown in Figure b. Therefore, the slicing technique alone does not solve this example. However, one can define two intermediate configurations (marked **1'** and **2'** in Figure c) which are close to the initial and goal configurations, respectively, such that the projection technique of Subsection 7.1 applies from **1'** to **2'**, and the slicing technique applies from **1** to **1'** and from **2'** to **2**.

space.

- Let us consider a slightly more complicated example, requiring the projection and slicing techniques to be combined. The robot is a planar arm with three revolute joints (see Figure 15). The initial and goal configurations of the arm, $\mathbf{q}_{init}$ and $\mathbf{q}_{goal}$, are marked **1** and **2** in the figure. Neither the projection technique of Subsection 7.1 applied to the third joint, nor the slicing technique applied by rigidifying the third joint allows us to plan a free path. Indeed, the initial (resp. goal) orientation of the third link cannot be achieved without collision at the goal (resp. initial) orientation of the first two links. However, the projection technique applied to the third joint makes it possible to generate a path from a free configuration $\mathbf{q}'_{init}$ (marked **1'** in the figure) close to $\mathbf{q}_{init}$, to a free configuration $\mathbf{q}'_{goal}$ (marked **2'**) close to $\mathbf{q}_{goal}$. Let $\mathcal{C}'$ be the configuration space of the first two links. $\mathbf{q}'_{init}$ (resp. $\mathbf{q}'_{goal}$) can be selected as a configuration that (1) projects in the free subspace of $\mathcal{C}'$ at a configuration

close to the projection of $\mathbf{q}_{init}$ (resp. $\mathbf{q}_{goal}$), and (2) produces the same orientation of the third link with respect to the second link as $\mathbf{q}_{init}$ (resp. $\mathbf{q}_{goal}$). Then, it remains to plan a free path from $\mathbf{q}_{init}$ to $\mathbf{q}'_{init}$, and another one from $\mathbf{q}'_{goal}$ to $\mathbf{q}_{goal}$. This can be done by rigidifying the third joint and using the slicing technique. In the particular case of Figure 15, these two paths are rotations of the second joint. The orientation of the third link is changed along the path from $\mathbf{q}'_{init}$ to $\mathbf{q}'_{goal}$. ■

If planning in a cross-section of configuration space produces a free path, then this path is a free path in $\mathcal{C}$. But, if planning fails in a cross-section, this does not imply that there is no free path in $\mathcal{C}$.

7.3 Simplification of the Shape of Objects

The complexity of motion planning increases with the number of polynomial constraints defining the free space and the maximal degree of these constraints. One way to simplify these constraints is to approximate the shapes of both the robot and the obstacles by reducing the number of surfaces defining their boundary and/or the degree of the equations of these surfaces.

Simple and often efficient techniques consist of approximating the objects by rectangloids or balls. One can also approximate objects by more sophisticated polygons or polyhedra. This reduces the maximal degree of the constraints, but if one is tempted to increase the accuracy of the approximation, it ultimately results in an increase in the number of curves/surfaces, i.e. the number of constraints.

One may usefully distinguish between two kinds of approximation: *bounding* and *bounded* ones. A bounding approximation consists of replacing objects by new ones which completely contain the original ones. If a free path is generated with such an approximation, it is also a free path with the original objects (but the converse is not true). A bounded approximation consists of replacing objects by new objects completely contained in the original ones. If no free path is generated with such an approximation, it implies that there is no free path with the original objects (again, the converse is not true). Thus, bounded approximation can be used to detect that a particular problem has no solution.

One can generalize the above ideas and consider several bounding and bounded approximations of increasing precision. Then the planner it-

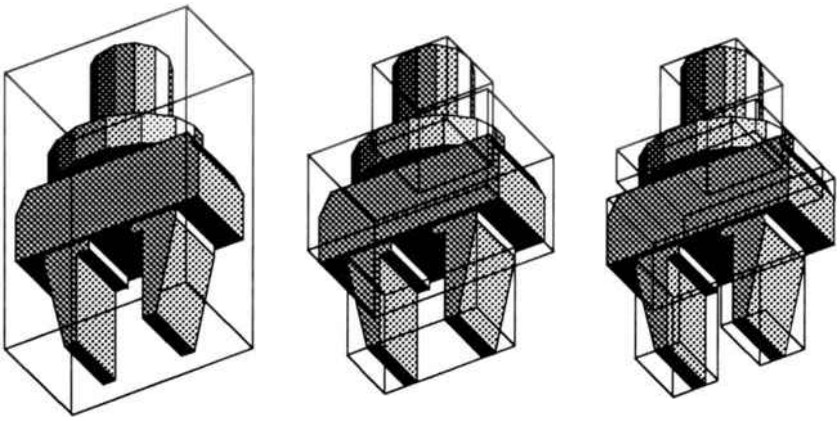


Figure 16. This figure shows three bounding approximations (from coarse to fine) of a robot gripper using rectangloids [Pasquier, 1989]. The motion planner may use such models in sequence until it finds a path.

eratively uses finer and finer approximations of the objects, until a free path is generated or there is evidence that no free path exists. Figure 16 illustrates a sequence of bounding approximations for a robot gripper.

7.4 Focusing Attention on a Subset of the Workspace

The techniques introduced in the three previous subsections are "simplification" heuristics. They exploit the fact that in most cases, motions take place in a relatively uncluttered space, so that there still remain solution paths after the various approximations have eliminated some. In many tasks, however, there are isolated localities that are quite cluttered and/or where contacts with obstacles have to be made (e.g. grasping, part mating). Simplification heuristics cannot be applied to these cluttered localities.

If cluttered localities can be identified in advance, one can break the original planning problem into subproblems. The subproblems involving cluttered localities only require a restricted subset of the workspace to be considered. Hence, the number of constraints on the motions is considerably reduced with respect to the original problem.

Example: Consider the problem of adding a part to an assembly with a robot arm. It can be broken down into five subproblems: (1) move the

arm to a configuration near the part, (2) grasp the part, (3) transfer the part to the current subassembly, (4) mate the part with the subassembly, and (5) move away to some rest configuration. In general, only subproblems (2) and (4) involve cluttered localities. In both subproblems, the motions usually have small amplitude, so that motion planning can be restricted to a small subset of the workspace. For instance, in (4), one can consider the part to be positioned (and perhaps also the gripper) as the robot, rather than the whole arm, and restrict the workspace of this object to a small volume around the assembly. If one wants to be sure that the generated plan is safe, the motion of the whole arm can then be checked for possible collision. In case a collision is possible, the planner would have to find another solution for (4). ∎

It can be further observed that in many cases cluttered localities are stereotyped situations (e.g. a mobile robot moving through a door or turning in a corridor). It may be worthwhile to attempt to recognize these situations and to apply specific (and more efficient) planning techniques to them. Along this line, Maddila and Yap explored the problems of moving a polygon around the corner in a corridor [Maddila and Yap, 1986] and through a door [Yap, 1987b]. Specific planning techniques applicable to stereotyped situations are called *local experts*.

8 Relation to Other Problems

An automatic motion planner can be used in a variety of ways. For instance, it can be included in an off-line robot programming system [Latombe et al., 1985] in order to simplify the task of describing robot trajectories. It can also be part of an interactive graphic process planning system (e.g. for planning the machining of a mechanical part, the assembly of an electromechanical device, or the construction of a power plant), and be used to check the geometrical feasibility of the planned operations (e.g. drilling a hole, mating two parts, erecting a pipe). More generally, it can be used to automatically generate graphic animated scenes. Nonetheless, beyond these "short-term" applications, the ultimate goal of automating motion planning is to create autonomous robots.

As mentioned earlier in this chapter, automatic path planning is a critical problem in the creation of autonomous robots, but it is not the only one. Other major problems include real-time motion control, sensing,

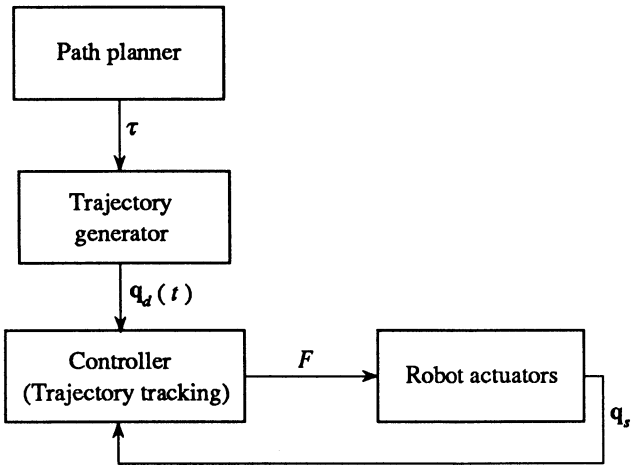


Figure 17. The path τ produced by the path planner is input into a trajectory generator that determines the time dependence of the configuration parameters. The output of the trajectory generator — the desired configurations q_d as a function of time — is fed into the controller. The controller computes the force F to be exerted by each actuator from the deviation of the current configuration q_s as measured by the sensors, relative to q_d.

and task-level planning. Because of the potentially complex interactions among them, the design of a truly autonomous robot will certainly require all these problems to be solved concurrently. But, today, our current understanding of the issues is insufficient to present a comprehensive picture of such a design. In the following subsections we therefore limit our presentation to a brief analysis of some possible interactions between motion planning and the three other problems mentioned above.

8.1 Interaction with Real-Time Motion Control

A motion plan specifies a motion to be executed by the robot. For example, a path specifies a continuous sequence of configurations that the robot should traverse toward the goal configuration. The basic task of the real-time controller is to make the robot execute the motion plan, e.g. follow the generated path.

Let us first assume that the motion plan is a free path τ. The task of

the controller is to transform τ into forces[5] to be exerted by the robot actuators. Typically, this transformation is broken down into two steps. The first step, called *trajectory generation*, consists of deciding on the velocity profile along the path. It can be done prior to motion execution. The second step, called *trajectory tracking*, consists of computing the forces to be exerted by the actuators at each time in order to perform the desired motion. The trajectory tracking step may use the dynamic equation of the robot — the equation expressing that the forces applied by the actuators equal the resultant of the various forces acting on the robot during motion, i.e. the gravitational, frictional, inertial, centrifugal and Coriolis forces — to compute the force that has to be delivered by each actuator. If the dynamic equation used by the controller was a perfect model, no feedback would be necessary. However, due to various disturbances, sensing is necessary to determine the deviation between the desired state and the actual state of the robot. While the motion is being executed, the controller computes the actuator forces which tend to eliminate this deviation at an update rate which typically ranges between 10 and 1000 Hz. Figure 17 shows the relationship between the path planner, the trajectory generator, the controller and the robot.

The simple architecture of Figure 17 is a classical one. Its advantage is to break the problem of deciding on a motion into clearly defined subproblems. But it may lead to rather inefficient motions. Even if the controller attempts to minimize execution time, the geometry of the paths may not allow the controller to fully exploit the capabilities of the actuators. A better approach is probably to take the dynamic equation of the robot, together with the actuators' saturation limits, into account during motion planning in order to generate paths allowing (quasi-)time-optimal motions to be executed. Several techniques have been recently proposed for implementing this approach [Donald and Xavier, 1989] [Barraquand, Langlois and Latombe, 1989a] [Shiller and Dubowsky, 1989] [Jacobs et al., 1989]. Such techniques may become particularly important when the robot operates in a dynamic workspace among mobile obstacles, when the task has to be accomplished within tight time bounds (e.g. because it interacts with an external process), and/or when the robot's productivity has to be optimized (e.g. in manufacturing).

[5]Here we use the word "force" in the generalized sense. It is either a force or a torque depending on the nature of the actuator.

TESLA EXHIBIT 1013, Page 44

Paths are the simplest sort of motion plans. We saw in Subsection 5.3 that in the presence of uncertainty, motion plans may include sensory-based motion commands, e.g. force-compliant motion commands, which are more tolerant to errors than classical position-controlled motion commands. The availability of such commands may make the task of the motion planner considerably simpler [Shekhar and Khatib, 1987]. The concurrent design of sensory-based motion primitives and motion planning methods using them would be a major step toward a better integration of motion planning and control. It has received little attention so far.

8.2 Interaction with Sensing

In the basic motion planning problem, we assumed that the robot has complete and exact knowledge of its environment. As mentioned in Subsection 5.3, this assumption is not always realistic. There are many real-life problems where the geometry of the workspace can only be partially known at planning time. Since planning requires the robot to have a model of its environment, the more incomplete this model, the more limited the role of planning.

If uncertainty in the prior knowledge is small, it is reasonable to anticipate all possible contingencies and to generate sensory-based motion plans that can deal with them (see Subsection 5.3). Sensing is used to guide the motions and monitor their execution. Planning makes sure that the motion commands will make enough information available at execution time so that the controller will be able to both monitor progress toward the goal and recognize goal achievement reliably.

On the other hand, if the robot has no prior knowledge about its environment, planning is useless. Then, the robot must rely heavily on sensing to acquire information while it is moving and it must react to the sensory data. For example, if the robot is equipped with proximity range sensors, a potential field method (see Subsection 4.3) can be applied on-line to guide the motion. The input goal configuration is used to create the attractive potential, while the data provided by the proximity sensors are used to create the repulsive potential. The negated gradient is treated as an external force, and the robot is controlled to comply to that force (i.e. to move along the flow of the negated gradient field) [Khatib, 1986]. There is no prior planning, and the robot can easily get

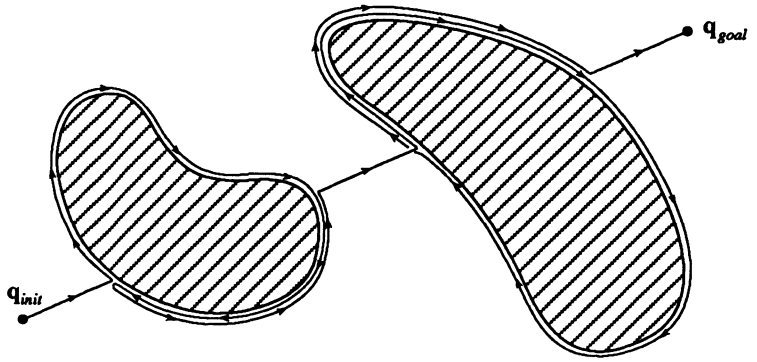


Figure 18. The robot has no prior knowledge of the obstacles. It follows the straight line segment joining q_{init} to q_{goal} until it hits an obstacle. Then it follows the C-obstacle's boundary, while keeping track of the intersections between this boundary and the line segment joining q_{init} and q_{goal}. When it comes back to the hitting point, the robot returns to the intersection point that is closest to q_{goal} by following the shortest path along the C-obstacle's boundary. From this point, it resumes moving along the straight line segment toward q_{goal}.

trapped into a local minimum of the potential field, e.g. a minimum created by an obstacle concavity. Escaping from such a local minimum requires some trial-and-error exploration by the robot.

Another reactive scheme has been proposed by Lumelsky when no prior knowledge is available [Lumelsky, 1987a]. The configuration space is assumed to be two-dimensional and the robot is equipped with a position sensor and a peripheral touch sensor that allows it to track the boundaries of the obstacles. Lumelsky described several specific algorithms which are variants of each other. One of them, illustrated in Figure 18, consists of having the robot move along a straight line toward the goal configuration q_{goal}. If the robot hits an obstacle, it shifts to moving along the boundary of the corresponding C-obstacle in a predetermined direction (e.g. clockwise) until it eventually comes back to the collision point. During this motion, it identifies the intersection point of the C-obstacle's boundary and the line segment joining the initial and goal configurations that is closest to the goal. After having made the closed-loop path around the C-obstacle it returns to this point by following the

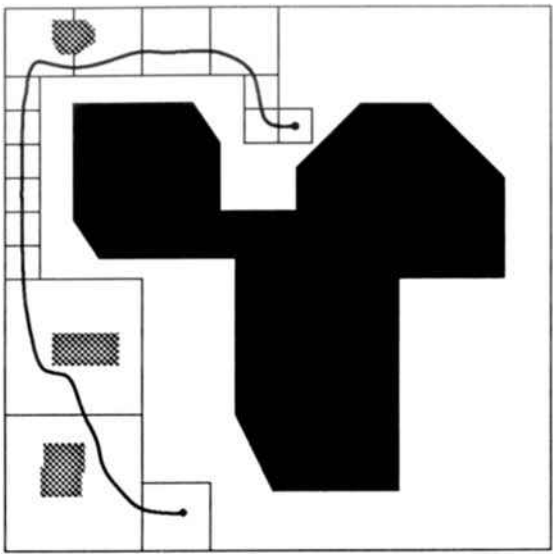


Figure 19. The region shown black is the C-obstacle region known at planning time. The regions shown grey depict unknown C-obstacles that are actually present. Knowledge available at planning time is used to compute a quadtree decomposition of the configuration space and extract a channel composed of adjacent cells lying outside the known C-obstacle region. At execution time, a sensory-based potential field is used to guide the robot through the channel away from the unexpected obstacles.

shortest path along the C-obstacle's boundary. From there it resumes moving along a straight line toward the goal. This simple algorithm is guaranteed to attain $\mathbf{q}_{goal}$ whenever every C-obstacle is bounded by a simple closed curve of finite length and any straight line segment intersects a finite number of C-obstacles. The length of the generated path is upper bounded by $D + 1.5\Sigma_i p_i$, where D is the distance between $\mathbf{q}_{init}$ and $\mathbf{q}_{goal}$ and the p_i's are the perimeters of the C-obstacles intersecting the disc of radius D centered at $\mathbf{q}_{init}$. Some steps toward generalizing this kind of algorithm to configuration spaces of dimensions greater than two have been made in [Sun and Lumelsky, 1989] and [Cheung and Lumelsky, 1990].

The extreme case where the robot has no prior knowledge of its environment is rare. A more realistic situation is when the robot knows the

shape and location of the largest objects in its workspace, but has incomplete knowledge about smaller objects. An approach proposed in [Choi, Zhu and Latombe, 1989] to deal with such a situation is the following:

- first, a cell decomposition method is used to plan a channel among the known C-obstacles between the initial and goal configurations.
- Second, a potential field collision-avoidance method is used to guide the robot in the channel at execution time.

The potential field combines a dynamic attractive potential, generated by a target configuration that moves toward the goal configuration while the robot progresses in the channel, a repulsive potential, generated by the boundary of the channel (so that the robot's configuration does not escape the channel), and a repulsive potential, computed from the data provided by proximity range sensors. Figure 19 illustrates this approach in a two-dimensional configuration space. In this example, the channel was generated from a quadtree decomposition (the known C-obstacle region is shown black). The C-obstacles shown grey were not known at planning time and were detected during the motion. As long as the unexpected obstacles are small and sparsely distributed, the robot is unlikely to get trapped in a local minimum. However, there is no guarantee that this will never happen. When a local minimum is encountered, some replanning may be necessary using an updated model of the workspace.

There is a wide variety of possible software architecture interweaving motion planning and execution when prior knowledge is incomplete. The "best" architecture depends on both the nature and the extent of the lack of knowledge.

8.3 Interaction with Task-Level Planning

A motion planning goal is specified as a spatial arrangement of physical objects. Achieving such a goal contributes to the achievement of a more global task. For instance, the robot may be a troubleshooter whose task is to diagnose faults in pieces of electromechanical equipment and to repair them. Finding a fault typically requires the equipment to be inspected and operated, measuring devices to be connected, and probes to be inserted and removed. Each such operation can be specified by a spatial arrangement of objects, but deciding which operations must be

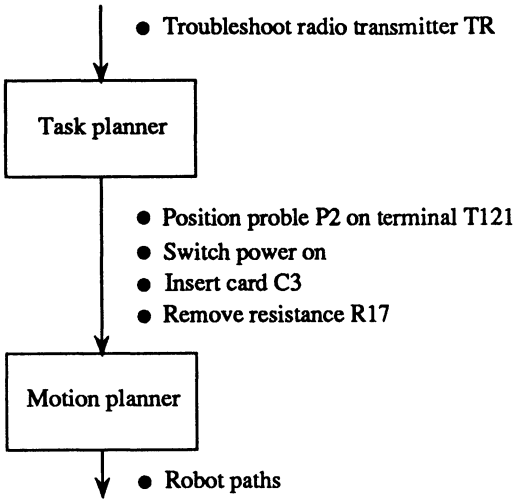


Figure 20. This figure illustrates the relationship between task planning and motion planning. The input to the task planner is a task description, e.g. "troubleshoot radio transmitter TR". Its output is a set of operations, e.g. "position proble P2 on terminal T121", "switch power on", "insert card C3", "remove resistance R17". Each operation can be specified as a goal spatial arrangement of objects (e.g. the goal position of the probe). The motion planner generates the motion plan for achieving this goal.

executed and their execution sequence — i.e. *task planning* — requires "higher-level" reasoning and planning capabilities than those possessed by a motion planner.

Although much research still remains to be done, Artificial Intelligence (AI) has produced a variety of methods for reasoning about tasks and for planning the high-level operations needed to accomplish them [Nilsson, 1980] [Genesereth and Nilsson, 1987] [Georgeff, 1987]. These methods are readily available to build planners whose outputs are sequences of operations, each specifying a new goal spatial relationship to the motion planner. Figure 20 illustrates the relationship between the task planner and the motion planner.

However, an AI task planner typically represents a task using logic-oriented constructs (e.g. predicate calculus language) which contain little information about the actual geometry of the objects and the spatial re-

lations among them. Therefore, a plan generated by the task planner may turn out to be infeasible when developed at the motion level. For instance, there may exist no path for a probe to the terminal at which it should be connected. In such a situation, the task planner would have to generate another plan. Important questions are: What kind of pertinent information should the motion planner feed back to the task planner? How can the task planner use this information to generate another plan?

In addition, the behavior of the equipment to be repaired might not be stationary, hence requiring the troubleshooting operations to be executed within strict time constraints. For instance, a probe may have to be moved from one position to another so that two measurements can be done within a short interval of time. However, the durations of the motions are typically not known with good precision until the motion plans have been generated.

This discussion suggests that integrating task planning and motion planning is a much more ambitious enterprise than Figure 20 suggests. Little has been done so far in this domain.

9 Literature Landmarks

We close this introductory chapter with a brief discussion of key publications that can be considered to be landmarks in the development of robot motion planning. The citations are given approximately in chronological order, so that this section may also be regarded as a short historical account of the development of the motion planning field over the past 20 years. Many other references will be given throughout the book.

Nilsson [Nilsson, 1969] described a mobile robot system (Shakey) with planning capabilities. Although most of the planning done by Shakey used a model of the workspace and the robot actions expressed in the predicate calculus language (Strips planner), Nilsson introduced the visibility graph method for planning Shakey's motions.

Ambler and Popplestone [Ambler and Popplestone, 1975] described a method for inferring the quantitative location of objects in terms of homogeneous coordinate matrices, from input spatial symbolic relations among features of objects. This method was extended later to a larger set of relations and implemented as part of a high-level robot program-

ming language (Rapt) [Popplestone, Ambler and Bellos, 1980].

Liebermann and Wesley [Liebermann and Wesley, 1977] and Lozano-Pérez [Lozano-Pérez, 1976] presented the first attempts to build integrated systems (Autopass and Lama, respectively) for automatically programming robot arms. The input to these systems was the description of a mechanical assembly, in the form of a set of geometrical models of the individual parts and goal assembly relations among the parts. The task of the systems was to automatically generate the robot programs for assembling the parts. Although these systems were never fully implemented, they have contributed in emphasizing the importance of geometrical reasoning in robot planning and in pointing out key motion planning problems in the context of mechanical assembly.

Taylor [Taylor, 1976] investigated the problem of planning motions in the presence of uncertainty. He proposed a method, known as "skeleton refinement", based on numerical uncertainty propagation techniques. A similar approach was also proposed in [Lozano-Pérez, 1976]. An improvement of the method, using symbolic propagation techniques rather than numerical ones, was later described by Brooks [Brooks, 1982].

Udupa [Udupa, 1977] introduced the idea of shrinking the robot to a point in an appropriate space (the term "configuration space" was not used yet). Lozano-Pérez and Wesley [Lozano-Pérez and Wesley, 1979] exploited this idea in a more systematic fashion, and proposed the first path planning algorithm for polygonal and polyhedral robots and obstacles without rotation. Their work is usually considered as the first contribution to "exact" motion planning. Later, Lozano-Pérez borrowed the notion of "configuration space" from Mechanics, and popularized it in motion planning. He also extended the ideas contained in [Lozano-Pérez and Wesley, 1979] and introduced the principle of the approximate cell decomposition approach [Lozano-Pérez, 1981 and 1983].

Reif [Reif, 1979] presented the first theoretical investigation of the inherent computational complexity of the path planning problem, showing that planning a free path in a configuration space of arbitrarily large dimension among fixed obstacles is PSPACE-hard. This work has contributed in attracting the interest of theoretical computer scientists toward motion planning. It has been followed by a flurry of papers on lower-bound complexity for a variety of motion planning problems (see

TESLA EXHIBIT 1013, Page 51

Section 6). Reif termed this area *Computational Robotics* [Reif, 1987].

Chatila [Chatila, 1982] investigated motion planning with incomplete knowledge for a mobile robot represented as a point in a two-dimensional workspace. He was the first to base his planner on an exact decomposition of the empty subset of the workspace into convex cells (an idea which was extended later to free space in configuration space). The planner was operating on-line, and the decomposition was periodically updated in order to take into account new information provided by the sensors.

Mason [Mason, 1982] stressed the importance of including pushing motions in a variety of manipulation tasks. Although his initial work was mainly aimed at modeling the mechanics of pushing under quasi-static assumptions, it has been followed by other publications by Mason and others directly related to motion planning.

Ó'Dúnlaing and Yap [Ó'Dúnlaing and Yap, 1982] introduced retraction as a new theoretical approach for path planning. Independently, and almost simultaneously, Brooks [Brooks, 1983a] presented a more empirical, "retraction-like" planning method, known as the freeway method.

In 1983 and 1984, Schwartz, Sharir, and Ariel-Sheffi published a series of five papers called the Piano Movers' Problem series [Schwartz and Sharir, 1983a, 1983b and 1983c] [Sharir and Ariel-Sheffi, 1983] [Schwartz and Sharir, 1983d]. The first paper of the series described the first exact method for planning free paths of a polygonal object allowed to both translate and rotate in a two-dimensional polygonal workspace. The second paper established the first upper bound on the time complexity of path planning in a semi-algebraic free space of any fixed dimension.

Laugier and Pertin [Laugier and Pertin, 1983] described an implemented method for planning the grasp position of a mechanical hand with two parallel fingers on an object bounded by planar, cylindrical, and spherical surfaces. In a first processing phase, the planner generates various grasps and assesses their local feasibility using a collection of empirical tests. In a second phase, it checks the global accessibility of each grasp by planning a path in the configuration space of the hand. These ideas were refined later by other authors.

Lozano-Pérez, Mason and Taylor [Lozano-Pérez, Mason and Taylor,

1984] described the preimage backchaining approach to motion planning with uncertainty in control and sensing. For the first time, this approach considered compliant motions in planning. In the original paper, the approach was essentially a theoretical framework, and was not implemented. Since then, the theory and the implementation of preimage backchaining have been investigated further.

Dufay and Latombe [Dufay and Latombe, 1984] described an implemented approach to motion planning with uncertainty based on inductive learning. The approach consists of executing the same task several times and combining the execution traces into a more general strategy. More recently, other researchers have investigated different learning techniques for motion planning and motion skill acquisition.

Gouzènes [Gouzènes, 1984a and 1984b] described the first implemented approximate cell decomposition method to plan the motion of robot arms with revolute joints. His method consists of approximating the free space as a collection of rectangloid cells obtained by dividing the range of possible angular positions of each link into small sub-intervals. The same type of method was later used and refined by several other researchers.

Khatib [Khatib, 1986] pioneered the potential field approach[6]. Although he implemented it as a real-time collision avoidance module in a robot controller, the approach is extendable to motion planning, which has been done since then. Koditschek [Koditschek, 1987] introduced the notion of a "navigation function", a local-minimum-free potential function. Using a variant of the potential field approach, Faverjon and Tournassoud [Faverjon and Tournassoud, 1987] implemented a practical system able to plan the motions of a manipulator arm with eight joints moving among vertical pipes. Barraquand and Latombe combined a potential field method with random techniques to escape from local minima and implemented a planner for generating free paths for robots with many degrees of freedom [Barraquand and Latombe, 1989a].

Laumond [Laumond, 1986 and 1987b] considered the problem of planning free paths for nonholonomic car-like mobile robots. He produced the interesting result that a free path for a free-flying robot in a two-

[6] Actually, Khatib introduced the principle of this approach in earlier publications (see [Khatib, 1980]). However, the paper cited here is the first comprehensive description of the approach.

dimensional workspace can always be transformed into a free path for a nonholonomic car-like robot having the same geometry and moving in the same workspace, by introducing simple backup maneuvers. Li and Canny [Li and Canny, 1989] were the first to apply tools developed in controllability theory for nonlinear systems to nonholonomic robots. These tools make it possible to generalize Laumond's results.

Lozano-Pérez et al. [Lozano-Pérez et al., 1987] described an implemented robot system, Handey, integrating a grasp planner and a path planner. This system is able to plan the motions of a manipulator robot for constructing simple assemblies made of polyhedral objects, and to execute the plans, assuming no uncertainty. Part of the initial knowledge about the workspace is obtained by using a vision system. In addition, the system is capable of planning the re-grasping of an object, if no grasp at the initial location of the object is compatible with a grasp at the goal location.

Exercises

1: Discuss in which respects motion planning differs from: (1) collision checking; (2) collision avoidance.

2: Design an algorithm for checking whether two polygons intersect.

3: Design an algorithm for checking whether two polyhedra intersect.

4: Consider the 3×3 matrix Θ defined in Subsection 3.1 for representing the orientation of a rigid object $\mathcal{A}$ in $\mathcal{W} = \mathbf{R}^3$. What equations relate the 9 parameters of the matrix? How many parameters are needed to represent a rotation? Comment on your results.

5: It is often said that representing the robot as a point in its configuration space transforms the problem of planning the motion of a dimensioned object into the "simpler" problem of planning the motion of a point. Do you agree or disagree with this assertion? Elaborate on your answer.

6: Let $\mathcal{C}$ be the configuration space of a compact rigid object $\mathcal{A}$ moving

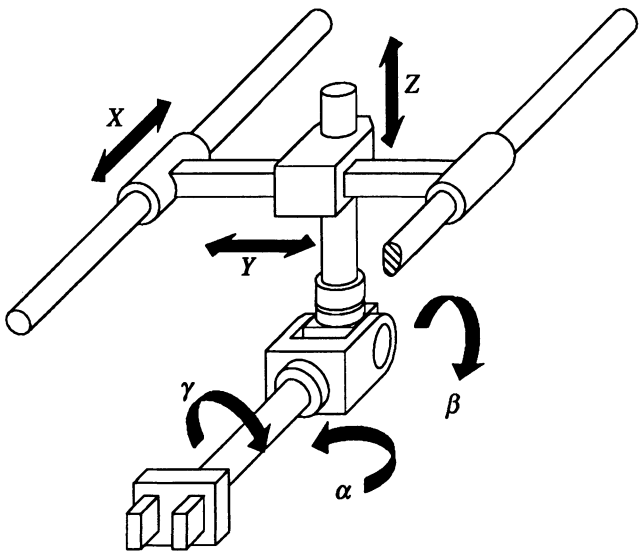


Figure 21. The first three joints of this robot are three perpendicular prismatic joints. The last three joints are revolute joints whose axes intersect at a single point.

freely in $\mathbf{R}^N$. Verify that the map $d : \mathcal{C} \times \mathcal{C} \to \mathbf{R}$ defined by:

$$\forall \mathbf{q}, \mathbf{q}' \in \mathcal{C} \ : \ d(\mathbf{q}, \mathbf{q}') = \max_{a \in \mathcal{A}} \|a(\mathbf{q}) - a(\mathbf{q}')\|,$$

where $\|x - x'\|$ denotes the Euclidean distance between two points x and x' in $\mathbf{R}^N$, determines a distance in $\mathcal{C}$.

7: Discuss the assumptions under which it is realistic to consider a wheeled mobile robot as a two-dimensional object moving freely in the plane.

8: Consider the simple six-joint gantry-type robot illustrated in Figure 21. The first three joints (denoted by X, Y and Z) are prismatic joints perpendicular to each other; the last three joints (denoted by α, β and γ) are revolute joints whose rotation axes intersect at a single point. Discuss the assumptions under which it is realistic to consider the end-effector of this robot as a free-flying object moving among obstacles in a three-dimensional workspace.

9: Consider the case where p free-flying objects $\mathcal{A}_1, ..., \mathcal{A}_p$ operate in the same workspace $\mathcal{W}$ among q fixed obstacles $\mathcal{B}_1, ..., \mathcal{B}_q$. Treat these moving objects as a multi-bodied robot $\mathcal{A} = \{\mathcal{A}_1, ..., \mathcal{A}_p\}$ whose configuration space is denoted by $\mathcal{C}$ (see Subsection 5.1). Specify the C-obstacles and the free space in $\mathcal{C}$ as we did for a single rigid object in Section 3.

10: Consider a point moving in $\mathbf{R}^2$. Assume that its motions are imperfectly controlled, but that its actual position is always within a distance less than a known constant ε from the position where it is desired to be. How must the obstacles be grown so that path planning is guaranteed to generate a path whose execution will be collision-free? Illustrate with an example the need for more sophisticated motion planning methods capable of generating sensory-based motion plans, rather than just paths.

11: Consider a part mating operation. What are the sources of errors that may prevent a position-controlled motion to execute the operation successfully? Is it reasonable to bound these errors by uncertainty intervals and to plan a robust motion plan as suggested in Subsection 5.3?

12: Propose and discuss one or two extensions of the basic motion planning problem not presented in Section 5.

13: Define (in a sentence) the words "kinematics" and "dynamics". [Hint: Look into a good dictionary.]

14: Identify and discuss briefly some typical motion planning problems for: (1) a mobile robot carrying objects in an office environment; (2) a space platform robot actuated by thrusters and equipped with two arms for assembling an orbital platform.

15: What does it mean for an algorithm to have time complexity $O(1)$? $n^{O(1)}$? $O(2^n)$? $2^{O(n)}$?

16: Sketch a few techniques for approximating objects by simple geometrical models and possibly reducing the complexity of motion planning.

17: Compare the projection and slicing techniques for reducing the di-

TESLA EXHIBIT 1013, Page 56

mension of configuration space (see Subsections 7.1 and 7.2). Are they *both* useful? Why?

18: Propose a software architecture for interweaving motion planning and motion execution when prior knowledge is incomplete.

19: Compare motion planning to the type of planning performed by an Artificial Intelligence planner such as Strips [Nilsson, 1980]. Describe, in general terms, a system integrating the two types of planning.

TESLA EXHIBIT 1013, Page 57

Chapter 2

Configuration Space of a Rigid Object

In Chapter 1 we introduced configuration space as a space in which the robot maps to a point. The mathematical structure of this space, however, is not completely straightforward, and deserves some specific consideration. The purpose of this chapter and the next one is to provide the reader with a general understanding of this structure when the robot is a rigid object not constrained by any kinematic or dynamic constraint. This chapter mainly focuses on topological and differential properties of the configuration space. More detailed algebraic and geometric properties related to the mapping of the obstacles into configuration space will be investigated in Chapter 3.

Although our primary goal at this stage is to provide the necessary background for investigating the basic motion planning problem, our presentation is substantially broader. Indeed, some concepts which are not directly related to this problem contribute to give an articulate vision of configuration space. These concepts may also help the reader to realize the depth of motion planning in general. Some other notions, which will be applied when we investigate extensions of the basic problem, are easily introduced here.

TESLA EXHIBIT 1013, Page 58

In Section 1 we define the configuration space $\mathcal{C}$ of a rigid object. In Section 2 we represent this space by "embedding" it in a Euclidean space $\mathbf{R}^M$. In Section 3 we discuss the local topological and differential structure of $\mathcal{C}$ and we show that $\mathcal{C}$ is a "smooth manifold", i.e. a space that is locally like $\mathbf{R}^m$, where m is the dimension of $\mathcal{C}$ ($m < M$). In Section 4 we discuss several ways of representing a configuration as a list of independent real parameters (e.g. stereographic projection, Euler angles, quaternion). In Section 5 we develop the notion of a path and we show how the global topology of $\mathcal{C}$ differs from that of $\mathbf{R}^m$. In Section 6 we present the notion of a tangent space, which makes it possible to talk about directions in configuration space. The topology of $\mathcal{C}$ is defined in Section 3 as the topology induced by the Euclidean metric in $\mathbf{R}^M$ (the space in which we previously embedded $\mathcal{C}$). In Section 7 we propose other metrics that induce the same topology. In Section 8 we present several particular cases that have been overlooked in the previous sections; some of them yield particularly simple configuration spaces. In Section 9 we discuss set-theoretic and topological properties of the image of the workspace obstacles in configuration space. Finally, in Section 10 we describe how forces map into configuration space.

A large part of this chapter is built upon elementary notions in Differential Geometry and Topology. These notions are developed in more depth in textbooks such as [Guillemin and Pollack, 1974] and [Spivak, 1979].

1 Definition

We consider a rigid object $\mathcal{A}$ — the robot — moving in a physical workspace $\mathcal{W}$ (see Figure 1). We represent $\mathcal{W}$ as the N-dimensional Euclidean space $\mathbf{R}^N$, where $N = 2$ or 3, equipped with a fixed Cartesian coordinate system, or frame, denoted by $\mathcal{F}_\mathcal{W}$. We represent $\mathcal{A}$ at a reference position and orientation as a *compact* subset of $\mathbf{R}^N$. A moving frame, $\mathcal{F}_\mathcal{A}$, is attached to $\mathcal{A}$ so that each point in the robot has fixed coordinates in $\mathcal{F}_\mathcal{A}$. The origins of $\mathcal{F}_\mathcal{W}$ and $\mathcal{F}_\mathcal{A}$ are denoted by $O_\mathcal{W}$ and $O_\mathcal{A}$, respectively. $O_\mathcal{A}$ is called the **reference point** of $\mathcal{A}$. We assume that $\mathcal{A}$ has no symmetry[1] — hence, in particular, it is not a single point.

Strictly speaking, $\mathcal{W}$ is a "physical space", while $\mathbf{R}^N$ is a "mathematical

[1] Particular cases induced by symmetry will be examined in Section 8.

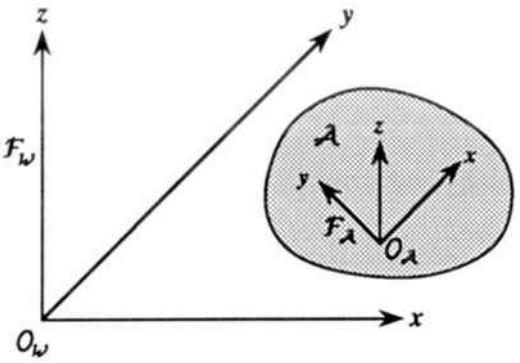


Figure 1. The robot $\mathcal{A}$ moves in a workspace $\mathcal{W} = \mathbf{R}^N$, with $N = 2$ or 3. $\mathcal{A}$ is modeled as a compact subset of $\mathbf{R}^N$. A fixed Cartesian frame $\mathcal{F}_\mathcal{W}$ is embedded in $\mathcal{W}$. A moving Cartesian frame $\mathcal{F}_\mathcal{A}$ is embedded in $\mathcal{A}$. The configuration of $\mathcal{A}$ specifies the position and orientation of $\mathcal{F}_\mathcal{A}$ with respect to $\mathcal{F}_\mathcal{W}$.

space" representing the former. However, we will not make this distinction any longer and we will use the same notation, $\mathcal{W}$, to denote both spaces.

The configuration space of $\mathcal{A}$ is defined as follows:

DEFINITION 1: *A* **configuration**[2] $\mathbf{q}$ *of* $\mathcal{A}$ *is a specification of the position and orientation of* $\mathcal{F}_\mathcal{A}$ *with respect to* $\mathcal{F}_\mathcal{W}$. *The* **configuration space** *of* $\mathcal{A}$ *is the space* $\mathcal{C}$ *of all the possible configurations of* $\mathcal{A}$. *A unique configuration of* $\mathcal{C}$ *is arbitrarily selected and is called the* **reference configuration** *of* $\mathcal{A}$. *It is denoted by* 0.

The configuration space $\mathcal{C}$ is intrinsically independent of the choice of the frames $\mathcal{F}_\mathcal{A}$ and $\mathcal{F}_\mathcal{W}$. Only the representation of $\mathcal{C}$ depends on these frames (see Section 2).

The subset of $\mathcal{W}$ occupied by $\mathcal{A}$ at configuration $\mathbf{q}$ is denoted by $\mathcal{A}(\mathbf{q})$. A point a of $\mathcal{A}$ is denoted by $a(\mathbf{q})$ in $\mathcal{W}$ when $\mathcal{A}$ is at configuration $\mathbf{q}$. Thus, for any two configurations $\mathbf{q}$ and $\mathbf{q}'$, $a(\mathbf{q})$ and $a(\mathbf{q}')$ are the same

[2]In some texts, the word "placement" is used instead of "configuration". The word "configuration" has been widely used in Mechanics for a long time. The use of the word "placement" seems more recent, with origins in Computational Geometry.

TESLA EXHIBIT 1013, Page 60

point in $\mathcal{A}$, but in general do not coincide in $\mathcal{W}$. More generally, if x is a geometric feature (e.g. a point, a vector, a set of points) in $\mathcal{A}$, $x(\mathbf{q})$ denotes the same feature in $\mathcal{W}$ when $\mathcal{A}$ is at configuration $\mathbf{q}$.

It is sometimes convenient to interpret a configuration as a rigid body transformation (also called a positive isometry), i.e. a transformation that preserves distance and orientation. The transformation $TR_{\mathbf{q}}$ corresponding to the configuration $\mathbf{q}$ in $\mathcal{C}$ rotates and translates $\mathcal{A}(0)$, i.e. the robot at its reference configuration, to $\mathcal{A}(\mathbf{q})$; we can write $TR_{\mathbf{q}}(\mathcal{A}(0)) = \mathcal{A}(\mathbf{q})$, where $TR_{\mathbf{q}}(x)$ denotes the result of applying $TR_{\mathbf{q}}$ to a geometrical feature x. For every $\mathbf{q} \in \mathcal{C}$, there exists a unique rotation r and translation t, such that $TR_{\mathbf{q}} = t \circ r$, where $t \circ r$ denotes the composition of t and r, with r applied first. $TR_{\mathbf{q}}$ is a continuous map of $\mathcal{W}$ onto itself. The space of rigid body transformations is known to have the structure of a non-commutative group, with the composition of two transformations as the internal binary operation.

2 Embedding in Euclidean Space

In this section we represent $\mathcal{C}$ as a subset of a Euclidean space $\mathbf{R}^M$, where M depends on the dimension N of $\mathcal{W}$. This space will be used in the next section to introduce the differential properties of $\mathcal{C}$. It will be called the *ambient space* of $\mathcal{C}$, and the representation of $\mathcal{C}$ will be referred to as the *embedding* of $\mathcal{C}$ in $\mathbf{R}^M$.

A configuration $\mathbf{q}$ of $\mathcal{A}$ can be bijectively represented, in a way that depends on the choice of the frames $\mathcal{F}_{\mathcal{A}}$ and $\mathcal{F}_{\mathcal{W}}$, as a pair $(\mathcal{T}, \Theta)$, where $\mathcal{T}$ determines the position of the origin $O_{\mathcal{A}}$ of $\mathcal{F}_{\mathcal{A}}$ in $\mathcal{F}_{\mathcal{W}}$ and Θ determines the orientation of $\mathcal{F}_{\mathcal{A}}$'s axes with respect to $\mathcal{F}_{\mathcal{W}}$.

Let us adopt the following convention: $\mathcal{T}$ is equal to the N-vector of the coordinates of $O_{\mathcal{A}}$ in $\mathcal{F}_{\mathcal{W}}$ and Θ is equal to the $N \times N$ matrix whose columns are the components of the unit vectors along $\mathcal{F}_{\mathcal{A}}$'s axes in $\mathcal{F}_{\mathcal{W}}$. Θ belongs to the so-called Special Orthogonal Group of the $N \times N$ matrices in $\mathbf{R}^{N^2}$ with orthonormal columns (and rows) and determinant $+1$. This group is denoted by $SO(N)$.

With the above convention, the configuration where $\mathcal{F}_{\mathcal{A}}$ and $\mathcal{F}_{\mathcal{W}}$ coincide corresponds to $\mathcal{T} = \vec{0}$ (null N-vector) and $\Theta = I$ (identity $N \times N$ matrix).

Choosing this configuration as the reference configuration 0, we have:

$$\forall a \in \mathcal{A} : \begin{cases} a & = & a(0) \\ a(\mathbf{q}) & = & \Theta\, a(0) + \mathcal{T} \end{cases}$$

where a denotes a point in $\mathcal{A}$ described by its N-dimensional coordinate vector in $\mathcal{F}_\mathcal{A}$ and $a(\mathbf{q})$ denotes the same point described by its coordinate vector in $\mathcal{F}_\mathcal{W}$ when $\mathcal{A}$ is at configuration $\mathbf{q} = (\mathcal{T}, \Theta)$.

The matrix Θ can be constructed as follows [Paul, 1981]:

- If $N = 2$:

$$\Theta = \begin{pmatrix} r_{11} & r_{12} \\ r_{21} & r_{22} \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}.$$

  This matrix represents the orientation of $\mathcal{F}_\mathcal{A}$ after a rotation of angle θ around $O_\mathcal{W}$, starting at the reference configuration.

- If $N = 3$:

$$\Theta = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix} =$$

$$\begin{pmatrix} n_x^2(1-\mathrm{c}\theta)+\mathrm{c}\theta & n_x n_y(1-\mathrm{c}\theta)-n_z\mathrm{s}\theta & n_x n_z(1-\mathrm{c}\theta)+n_y\mathrm{s}\theta \\ n_x n_y(1-\mathrm{c}\theta)+n_z\mathrm{s}\theta & n_y^2(1-\mathrm{c}\theta)+\mathrm{c}\theta & n_y n_z(1-\mathrm{c}\theta)-n_x\mathrm{s}\theta \\ n_x n_z(1-\mathrm{c}\theta)-n_y\mathrm{s}\theta & n_y n_z(1-\mathrm{c}\theta)+n_x\sin\theta & n_z^2(1-\mathrm{c}\theta)+\mathrm{c}\theta \end{pmatrix}$$

  where $\mathrm{c}\theta = \cos\theta$ and $\mathrm{s}\theta = \sin\theta$. This matrix represents the orientation of $\mathcal{F}_\mathcal{A}$ after a rotation of angle θ starting at the reference configuration and performed around an axis passing through $O_\mathcal{W}$ and oriented along a unit vector $\vec{n}$ whose components in $\mathcal{F}_\mathcal{W}$ are n_x, n_y, and n_z.

In the following, $\mathcal{M}_\mathcal{C}$ will denote the subset $\mathbf{R}^N \times SO(N) \subset \mathbf{R}^M$, with $M = N + N^2$. $\mathcal{M}_\mathcal{C}$ represents $\mathcal{C}$ in $\mathbf{R}^M$.

In the case where $N = 2$, $\mathcal{M}_\mathcal{C}$ is a subset of $\mathbf{R}^6$, with each configuration $\mathbf{q} \in \mathcal{C}$ identified by 6 real numbers $(x, y, r_{11}, r_{12}, r_{21}, r_{22}) \in \mathcal{M}_\mathcal{C}$. The constraints defining $\mathcal{M}_\mathcal{C}$ are the three polynomial equations:

$$\begin{aligned} h_1(x, y, r_{11}, ..., r_{22}) &= r_{11}^2 + r_{21}^2 - 1 = 0 \\ h_2(x, y, r_{11}, ..., r_{22}) &= r_{12}^2 + r_{22}^2 - 1 = 0 \\ h_3(x, y, r_{11}, ..., r_{22}) &= r_{11}r_{12} + r_{21}r_{22} = 0 \end{aligned}$$

expressing that the matrix:

$$\begin{pmatrix} r_{11} & r_{12} \\ r_{21} & r_{22} \end{pmatrix}$$

is orthonormal, and the polynomial equation:

$$r_{11}r_{22} - r_{21}r_{12} - 1 = 0$$

expressing that its determinant is $+1$. The first three equations define two disjoint subsets in $\mathbf{R}^6$ corresponding to matrices with determinants $+1$ and -1, respectively. The fourth equation merely selects the first of these two subsets.

In the case where $N = 3$, $\mathcal{M}_\mathcal{C}$ is a subset of $\mathbf{R}^{12}$, with each configuration identified by 12 numbers $(x, y, z, r_{11}, ..., r_{33}) \in \mathcal{M}_\mathcal{C}$. In the same fashion as above, expressing that the r_{ij}'s are elements of an orthonormal matrix 3×3 leads to six polynomial equations, $h_k(x, y, z, r_{11}, ..., r_{33}) = 0$, $k = 1, ..., 6$, which determine two disjoint subsets of $\mathbf{R}^{12}$. These two subsets again correspond to matrices with determinants $+1$ and -1, respectively. The additional constraint that the determinant be $+1$ selects one subset.

Notice that $SO(N)$ is a compact subset of $\mathbf{R}^{N^2}$. Indeed, it is closed, since it is the intersection of a finite number of closed sets. It is also bounded, since each element r_{ij} is contained in the interval $[-1, +1]$.

3 Manifold Structure

We now know how to represent the configuration space $\mathcal{C}$ of a rigid object $\mathcal{A}$ as the subset $\mathcal{M}_\mathcal{C} = \mathbf{R}^N \times SO(N)$ of $\mathbf{R}^M$, with $M = N + N^2$. In this section we show that $\mathcal{C}$ is locally "like" $\mathbf{R}^m$, for some fixed m depending on M. More precisely, every element of $\mathcal{C}$ has an open neighborhood that is diffeomorphic to an open subset of $\mathbf{R}^m$. This gives $\mathcal{C}$ the structure of a smooth m-dimensional manifold.

3.1 $\mathcal{C}$ is a Manifold

Let the topology of $\mathcal{C}$ be the subset topology induced by the Euclidean metric topology in $\mathbf{R}^M$, i.e. $X \subseteq \mathcal{M}_\mathcal{C}$ represents an open subset of $\mathcal{C}$ if and only if there exists an open subset Y of $\mathbf{R}^M$ such that $X = Y \cap \mathcal{M}_\mathcal{C}$.

Although this topology derives from the Euclidean metric in $\mathbf{R}^M$, it is considerably more general and can be induced by other metrics (see Section 7), including metrics which are independent of the representation of $\mathcal{C}$, such as the following distance function:

$$d(\mathbf{q}, \mathbf{q}') = \max_{a \in \mathcal{A}} \|a(\mathbf{q}) - a(\mathbf{q}')\|$$

where $\|x - x'\|$ denotes the Euclidean distance between any two points x and x' in a Euclidean space. All these metrics share the following "natural" property: The distance between two configurations $\mathbf{q}$ and $\mathbf{q}'$ tends toward zero when the regions $\mathcal{A}(\mathbf{q})$ and $\mathcal{A}(\mathbf{q}')$ tend to coincide, and conversely. In the following, we will consider the topology of $\mathcal{C}$ defined above independently of any particular metric.

The subset $\mathcal{M}_\mathcal{C}$ of $\mathbf{R}^M$ is defined by $\frac{1}{2}N(N+1)$ polynomial equations $h_k(u_1, u_2, ..., u_M) = 0$, with $k = 1$ to $\frac{1}{2}N(N+1)$, where $u_1, u_2, ..., u_M$ are merely new symbols for $x, ..., r_{11}, ..., r_{NN}$. (Actually, there is an additional equation: the equation that selects the matrices with determinant $+1$; this equation, however, has no effect on the local properties of $\mathcal{M}_\mathcal{C}$.) One can verify that, whether $N = 2$ or 3, the smooth functions h_k admit gradient vectors

$$\vec{\nabla} h_k = \left(\frac{\partial h_k}{\partial u_i}\right)_{i=1,...,M}$$

which are linearly independent at every $(u_1, ..., u_M) \in \mathcal{M}_\mathcal{C}$. In other words, $\mathcal{M}_\mathcal{C}$ is at the intersection of $\frac{1}{2}N(N+1)$ hypersurfaces of dimension $M-1$, such that at every intersection point, the tangent planes of these surfaces are distinct.

Let $m = M - \frac{1}{2}N(N+1) = \frac{1}{2}N(N+1)$. Every point $\mathbf{u} = (u_1, ..., u_M)$ in $\mathcal{M}_\mathcal{C}$ admits an open neighborhood U diffeomorphic to an open subset X of $\mathbf{R}^m$. Indeed, according to the Implicit Function Theorem (see Appendix A), there exists an open neighborhood U of $\mathbf{u}$ in $\mathcal{M}_\mathcal{C}$ with the following property: $\frac{1}{2}N(N+1) = M - m$ variables u_i, say $u_{m+1}, ..., u_M$, can be expressed as smooth functions g_i, with $i = m+1, ..., M$, of the m other variables $u_1, ..., u_m$; these functions are unique. The map $\phi : U \subset \mathcal{M}_\mathcal{C} \to \mathbf{R}^m$ defined by:

$$\phi(u_1, ..., u_M) = (u_1, ..., u_m)$$

is a smooth one-to-one map of U onto an open subset X of $\mathbf{R}^m$. In X,

it admits a smooth inverse $\phi^{-1} : X \subseteq \mathbf{R}^m \rightarrow U \subset \mathcal{M}_\mathcal{C}$, defined by:

$$\phi^{-1}(u_1, ..., u_m) = (u_1, ..., u_m, g_{m+1}(u_1, ..., u_m), ..., g_M(u_1, ..., u_m)).$$

Hence, ϕ is a diffeomorphism (see Appendix A).

A subset of $\mathbf{R}^M$ is a smooth m-dimensional manifold if and only if it is locally diffeomorphic to $\mathbf{R}^m$. By extension, any set that is related to this subset by a bijective map is also a smooth m-dimensional manifold. Thus:

PROPOSITION 1: *The configuration space $\mathcal{C}$ of a rigid object $\mathcal{A}$ moving in an N-dimensional workspace is a smooth manifold of dimension $m = \frac{1}{2}N(N+1)$. The bijective application that maps any configuration of $\mathcal{C}$ to its representation in $\mathcal{M}_\mathcal{C} \subset \mathbf{R}^M$, with $M = N(N+1)$, is called an* **embedding** *of $\mathcal{C}$ in $\mathbf{R}^M$. $\mathbf{R}^M$ is called the* **ambient** *Euclidean space.*

That $\mathcal{C}$ is a manifold[3] means that its local topological and differential structures are identical to those of $\mathbf{R}^m$. It is directly related to the well-known fact that infinitesimal rotations (equivalently, angular velocities) form a vector space [Paul, 1981].

The local likeness of $\mathcal{C}$ and $\mathbf{R}^m$ is emphasized by the standard definitions given in the following subsection.

3.2 Charts and Atlas

Let $u : \mathcal{C} \rightarrow \mathcal{M}_\mathcal{C}$ be the bijective application that maps any configuration $\mathbf{q}$ in $\mathcal{C}$ to its representation $\mathbf{u}$ in $\mathcal{M}_\mathcal{C} = \mathbf{R}^N \times SO(N)$.

A pair (U, ϕ) consisting of an open subset U of $\mathcal{C}$ and a diffeomorphism ϕ of the image of U in $\mathcal{M}_\mathcal{C}$ onto X, an open subset of $\mathbf{R}^m$, is called a **chart** or a **coordinate system** on U. For any $\mathbf{q}$ in U, let us write $\phi(u(\mathbf{q})) = (x_1(u(\mathbf{q})), ..., x_m(u(\mathbf{q})))$. The m smooth functions $x_1, ..., x_m$ are called the **coordinate functions** or more simply the **coordinates** of $\mathbf{q}$ in the chart. The inverse diffeomorphism ϕ^{-1} is called a **parameterization** of U. In the following, we will simply write $\phi(\mathbf{q}) = (x_1(\mathbf{q}), ..., x_m(\mathbf{q}))$.

Consider two charts (U, ϕ) and (V, ψ) such that $U \cap V \neq \emptyset$ (see Figure 2). Let X and Y be the images of U and V by ϕ and ψ, respectively. The

[3] From now on, we will omit the word "smooth" and simply say that $\mathcal{C}$ is a manifold.

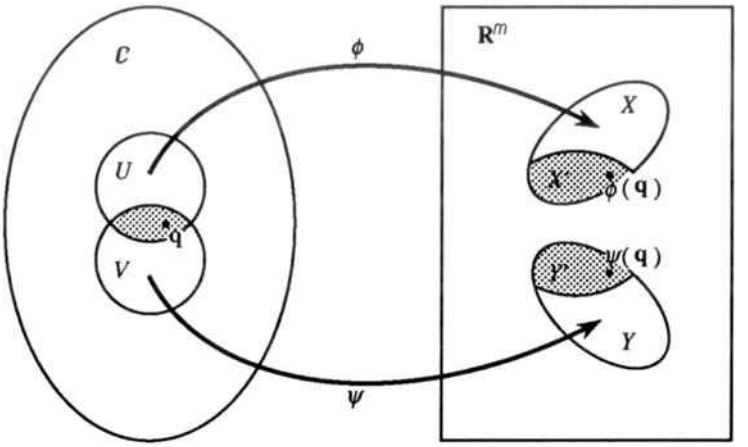


Figure 2. Two intersecting open subsets of $\mathcal{C}$, U and V, are mapped diffeomorphically onto two open subsets X and Y of $\mathbf{R}^m$ by ϕ and ψ, respectively. X' and Y' denote the images of $U \cap V$ by ϕ and ψ. The composed applications $\psi \circ \phi^{-1} : X' \rightarrow Y'$ and $\phi \circ \psi^{-1} : Y' \rightarrow X'$ are smooth, and the charts (U, ϕ) and (V, ψ) are said to be C^∞-related.

intersection $U \cap V$ is open and thus has two different charts $(U \cap V, \phi)$ and $(U \cap V, \psi)$. Let $X' \subseteq X$ and $Y' \subseteq Y$ be the images of $U \cap V$ by ϕ and ψ, respectively. For every $\mathbf{q} \in U \cap V$, we write:

$$\phi(\mathbf{q}) = (x_1(\mathbf{q}), ..., x_m(\mathbf{q})),$$

$$\psi(\mathbf{q}) = (y_1(\mathbf{q}), ..., y_m(\mathbf{q})).$$

ϕ has a smooth inverse ϕ^{-1}, so that the composed application $\psi \circ \phi^{-1} : X' \rightarrow Y'$, with:

$$\psi \circ \phi^{-1}(x_1(\mathbf{q}), ..., x_m(\mathbf{q})) = (y_1(\mathbf{q}), ...y_m(\mathbf{q})),$$

is smooth. In the same way, the inverse application $\phi \circ \psi^{-1} : Y' \rightarrow X'$ is also smooth. The charts (U, ϕ) and (V, ψ) are said to be **C^∞-related.**

A set of charts $\{(U_\alpha, \phi_\alpha)\}$ whose domains U_α cover $\mathcal{C}$ is called an **atlas** of $\mathcal{C}$. Once an atlas has been defined for $\mathcal{C}$, we can represent $\mathcal{C}$ as a collection of copies of $\mathbf{R}^m$ without referring to its embedding in $\mathbf{R}^M$. Since $\mathbf{R}^N$ can be covered by a single chart, and $SO(N)$ is compact, $\mathcal{M}_\mathcal{C}$,

hence $\mathcal{C}$, can be covered by a finite number of charts. (We recall that in a compact set $\mathcal{E}$, any infinite collection of open subsets covering $\mathcal{E}$ contains a finite subcollection covering $\mathcal{E}$.)

The fact that the charts in an atlas are C^∞-related will allow us to extend the differential properties established in a chart of the atlas to all the other charts.

3.3 Other Embeddings

As we mentioned above, we can consider $\mathcal{C}$ as an abstract space independently of the ambient space $\mathbf{R}^M$. The latter then looks somewhat arbitrary. Indeed, we could now pick any space $\mathbf{R}^{M'}$, with $M' > M$, and easily represent $\mathcal{C}$ as a subset $\mathcal{M}'_{\mathcal{C}}$ of $\mathbf{R}^{M'}$ equivalent (i.e. globally diffeomorphic) to $\mathcal{M}_{\mathcal{C}}$. $\mathbf{R}^{M'}$ would then be another ambient space of $\mathcal{C}$.

Since M is significantly greater than m, a legitimate question is the following: Does there exist $M'' < M$ such that $\mathbf{R}^{M''}$ contains a globally diffeomorphic copy $\mathcal{M}''_{\mathcal{C}}$ of $\mathcal{M}_{\mathcal{C}}$? A partial answer to this question is given by the Whitney Embedding Theorem: Every r-dimensional manifold can be embedded in $\mathbf{R}^{2r}$ [Guillemin and Pollack, 1974]. In other words, there is enough "twisting room" in $\mathbf{R}^{2r}$ for embedding any r-dimensional manifold. (Although the Whitney Theorem is optimal, it does not mean that a particular r-dimensional manifold cannot be embedded in a Euclidean space of dimension lower than $2r$; for instance, $\mathbf{R}^r$ is embedded in itself.)

Therefore, when we embed the one-dimensional manifold $SO(2)$ in $\mathbf{R}^4$ and the three-dimensional manifold $SO(3)$ in $\mathbf{R}^9$, there is some extra twisting room in the ambient spaces, which we do not use. We should at least be able to embed these manifolds in two- and six-dimensional Euclidean spaces, respectively.

Matrices in $SO(2)$ are of the form:

$$\begin{pmatrix} r_{11} & r_{12} \\ r_{21} & r_{22} \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} = \begin{pmatrix} r_{11} & -r_{21} \\ r_{21} & r_{11} \end{pmatrix}.$$

$SO(2)$ can be mapped to the unit circle in $\mathbf{R}^2$, denoted by S^1, with the following mapping:

$$(r_{ij})_{i,j\in[1,2]} \in \mathbf{R}^4 \mapsto (r_{11}, r_{21}) \in \mathbf{R}^2.$$

This mapping is a smooth bijective mapping whose inverse is also smooth. Hence, $SO(2)$ and S^1 are diffeomorphic, and S^1 can be regarded as an embedding of $SO(2)$ in $\mathbf{R}^2$. S^1 is so easy to visualize as a unit circle that we will often use S^1 in place of $SO(2)$.

Matrices in $SO(3)$ can be mapped to vectors in $\mathbf{R}^6$ by selecting the first two columns, i.e. :

$$(r_{ij})_{i,j\in[1,3]} \in \mathbf{R}^9 \mapsto (r_{11}, r_{21}, r_{31}, r_{12}, r_{22}, r_{32}) \in \mathbf{R}^6.$$

This mapping is smooth. In addition, since the third column can be obtained as the outer product of the first two, it is bijective, and its inverse is also smooth. Thus, it allows us to embed $SO(3)$ in $\mathbf{R}^6$. However, in the rest of the book, we will not use this embedding.

4 Parameterizations

In this section we describe various ways to chose charts and to construct an atlas of $\mathcal{C}$. Once an atlas has been constructed, every configuration can be represented by a vector of m parameters.

Since the parameterization of $\mathbf{R}^N$ is trivial, we only consider $SO(N)$. We first treat the case where $N = 2$, then the case where $N = 3$. Several parameterizations other than those presented below can be found in textbooks on Kinematics (e.g. [Bottema and Roth, 1979]) and Robotics (e.g. [Paul, 1981] [Craig, 1986]).

4.1 Parameterizations of $SO(2)$

A straightforward parameterization of $SO(2)$ consists of representing an orientation of the robot $\mathcal{A}$ by a single angle θ, say the angle between the x-axes of $\mathcal{F}_\mathcal{W}$ and $\mathcal{F}_\mathcal{A}$. The map is obviously not a one-to-one map, since all the angles $\theta + 2k\pi$, with $k \in \mathbf{Z}$, represent the same orientation of $\mathcal{A}$. One way to get a chart is to restrict the values of θ to an open interval of length 2π. Two such charts, say $0 < \theta < 2\pi$ and $-\pi < \theta' < +\pi$, are needed to have an atlas. However, in most path planning algorithms, it is sufficient to consider a single interval closed at one end, say $\theta \in [0, 2\pi)$, together with modulo 2π arithmetic on θ; but this parameterization is not a chart.

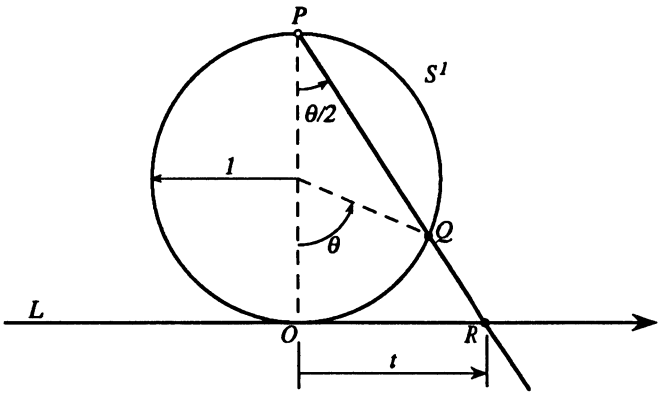


Figure 3. The stereographic projection allows us to map algebraically the circle S^1 onto the line of the reals (see text). It defines a chart valid over all S^1, except the point P. Two stereographic charts with two different points P are necessary and sufficient to build an atlas of S^1, or equivalently, $SO(2)$.

Some path planning methods require motion constraints to be expressed in algebraic form. Using the above parameterization yields expressions depending on θ through cosine and sine functions, i.e. non-algebraic functions. Another parameterization of $SO(2)$, which allows to express constraints in algebraic form, is known as the **stereographic projection.** It consists of projecting the whole unit circle S^1 (with the exception of one point) onto a line L tangent to the circle. L represents the set $\mathbf{R}$, with the origin at the point of tangency O (see Figure 3). Another line is drawn from the point P on the circle diametrically opposite the point of tangency. It intersects the circle at Q and L at R. The stereographic projection maps every point Q on the circle, except P, to the point R on L. In every open subset of S^1 not including P, the map is diffeomorphic, and hence defines a chart. Two stereographic charts, built with two different points P, are needed to form an atlas of S^1.

Let t be the abscissa of R along L and O correspond to $\theta = 0 \pmod{2\pi}$. We have:

$$t = 2 \tan \frac{\theta}{2}.$$

Writing $w = \frac{t}{2}$, we can express $\cos\theta$ and $\sin\theta$ as the following rational

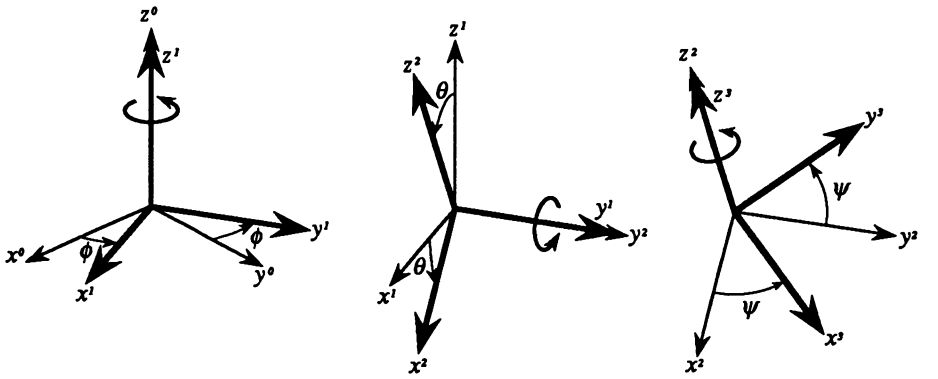


Figure 4. $\mathcal{F}_{\mathcal{A}}^{i} = (O_{\mathcal{A}}, x^i, y^i, z^i)$, $i = 0, 1, 2, 3$, denotes the frame $\mathcal{F}_{\mathcal{A}}$ at four successive orientations of $\mathcal{A}$. (ϕ, θ, ψ) are the Euler angles defining the orientation of $\mathcal{F}_{\mathcal{A}}^{3}$ relative to $\mathcal{F}_{\mathcal{A}}^{0}$. First, $\mathcal{A}$ is rotated by ϕ around the z^0-axis. Second, $\mathcal{A}$ is rotated by θ around the y^1-axis. Third, $\mathcal{A}$ is rotated by ψ around the z^2-axis.

fractions:

$$\cos\theta = \frac{1-w^2}{1+w^2} \quad \text{and} \quad \sin\theta = \frac{2w}{1+w^2}$$

leading to algebraic expressions in t of the constraints applying to $\mathcal{A}$'s motion.

There is no way to build an atlas of $SO(2)$ containing a single chart.

4.2 Euler Angles

The **Euler angles** are a widely used parameterization of $SO(3)$. They consist of three angles (ϕ, θ, ψ). The orientation of $\mathcal{A}$ they specify is obtained by applying three successive rotations to $\mathcal{A}$, starting at its reference orientation. Let $\mathcal{F}_{\mathcal{A}}^{0} = (O_{\mathcal{A}}, x^0, y^0, z^0)$ denote the frame $\mathcal{F}_{\mathcal{A}}$ at its reference orientation (see Figure 4). The rotations defined by ϕ, θ, and ψ are the following:[4]

- First, $\mathcal{A}$ is rotated by ϕ around the z^0-axis. Let $\mathcal{F}_{\mathcal{A}}^{1} = (O_{\mathcal{A}}, x^1, y^1, z^1)$ denote the frame $\mathcal{F}_{\mathcal{A}}$ at its new orientation.

[4]Slightly different specifications of these rotations are proposed in some textbooks. They simply determine different charts on $SO(3)$.

- Second, $\mathcal{A}$ is rotated by θ around the y^1-axis. Let $\mathcal{F}_{\mathcal{A}}^2 = (O_{\mathcal{A}}, x^2, y^2, z^2)$ denote the frame $\mathcal{F}_{\mathcal{A}}$ at its new orientation.

- Third, $\mathcal{A}$ is rotated by ψ around the z^2-axis. Let $\mathcal{F}_{\mathcal{A}}^3 = (O_{\mathcal{A}}, x^3, y^3, z^3)$ denote the frame $\mathcal{F}_{\mathcal{A}}$ at its new orientation.

The orientation of $\mathcal{F}_{\mathcal{A}}^3$ is the orientation of $\mathcal{A}$ defined by the Euler angles (ϕ, θ, ψ).

The parameterization of $SO(3)$ using (ϕ, θ, ψ) is not a one-to-one mapping. The same orientation is represented by $(\phi + 2k_1\pi, \theta + 2k_2\pi, \psi + 2k_3\pi)$, for any $k_1, k_2, k_3 \in \mathbf{Z}$. As for $SO(2)$, we may be tempted to apply modulo 2π arithmetic to every Euler angle. However, it is not so simple. Indeed, (ϕ, θ, ψ) represents the same orientation as $(\phi, \theta + \pi, 2\pi - \psi)$, so that a unique representation requires us to restrict the values of the three Euler angles to, say, $\phi \in [0, 2\pi)$, $\theta \in [0, \pi)$, and $\psi \in [0, 2\pi)$. In addition, even with this restriction, there still exists a singularity at $\theta = 0$. Indeed, if $\theta = 0$, the first and third rotations are made around the same axis. Thus, any two pairs (ϕ_1, ψ_1) and (ϕ_2, ψ_2), such that $\phi_1 + \psi_1 = \phi_2 + \psi_2$, determine the same orientation of $\mathcal{A}$. This means that in the $\phi\theta\psi$-space, a motion along a line $\phi = -\psi + c \pmod{2\pi}$ at $\theta = 0$, where c is a constant, corresponds to no motion of $\mathcal{A}$.

A single set of Euler angles (ϕ, θ, ψ), with appropriate restrictions on the values of ϕ, θ and ψ, is called an Eulerian chart. Several Eulerian charts can be defined by applying three successive rotations to $\mathcal{A}$ around different axes. However, no atlas on $\mathcal{C}$ can be constructed with Eulerian charts only, since the null rotation corresponds to a singularity in all these charts. When Euler angles are used to represent the robot's orientation in a motion planner, it is usually appropriate to use a single set of Euler angles $0 \leq \phi < 2\pi$, $0 \leq \theta < \pi$, $0 \leq \psi < 2\pi$, taking precautions at the interval extremities. If the singularities cannot be tolerated, it is preferable to use the quaternion parameterization presented in the next subsection.

Another drawback of Euler angles is that they do not allow us to express motion constraints in algebraic form. Applying a rotation specified by (ϕ, θ, ψ) to a geometric feature, say a point x, is equivalent to multiplying the coordinate vector of x by the corresponding matrix of $SO(3)$. The new coordinates of x are non-algebraic functions of ϕ, θ, and ψ. Hence,

some motion planning methods which require the motion constraints to be algebraic cannot make use of this parameterization.

The matrix $\mathbf{E}_{\phi\theta\psi}$ of $SO(3)$ corresponding to the three Euler angles defined above, with the conventions taken in Section 2, is:

$$\begin{pmatrix} \cos\phi & -\sin\phi & 0 \\ \sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{pmatrix} \begin{pmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{pmatrix} =$$

$$\begin{pmatrix} \cos\phi\cos\theta\cos\psi - \sin\phi\sin\psi & -\cos\phi\cos\theta\sin\psi - \sin\phi\cos\psi & \cos\phi\sin\theta \\ \sin\phi\cos\theta\cos\psi + \cos\phi\sin\psi & -\sin\phi\cos\theta\sin\psi + \cos\phi\cos\psi & \sin\phi\sin\theta \\ -\sin\theta\cos\psi & \sin\theta\sin\psi & \cos\theta \end{pmatrix}.$$

By restricting ϕ and ψ to $[0, 2\pi)$ and θ to $(0, \pi)$, and given a matrix of $SO(3)$:

$$\begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix}$$

the three Euler angles (ϕ, θ, ψ) are determined by the following relations:

$$\begin{aligned} \phi &= atan2(r_{23}, r_{13}) \\ \theta &= atan2(\sqrt{r_{31}^2 + r_{32}^2}, r_{33}) \\ \psi &= atan2(r_{32}, -r_{31}) \end{aligned}$$

where $atan2(y, x)$ is defined as the argument of the complex number $x + \vec{i}y$.

4.3 Quaternions

Quaternions encode orientations in a three-dimensional workspace with four parameters.[5]

A quaternion $\mathbf{X}$ is an element of $\mathbf{R}^4$ which can be split into a scalar part x_0 and a 3-vector $\vec{x}$. To simplify notation, the scalar part and the vector part are combined using the addition symbol, i.e. $\mathbf{X} = x_0 + \vec{x}$. It

[5]The theory of quaternions is described in many textbooks. The most complete reference is [Hamilton, 1969].

TESLA EXHIBIT 1013, Page 72

is convenient to treat 3-vectors as quaternions having a zero scalar part, and scalars as quaternions having a zero vector part.

The product of two quaternions $\mathbf{P}$ and $\mathbf{Q}$ is the quaternion $\mathbf{R} = \mathbf{PQ}$ specified by:

$$\mathbf{R} = r_0 + \vec{r}$$

with:

$$r_0 = p_0 q_0 - \vec{p} \cdot \vec{q}$$

$$\vec{r} = p_0 \vec{q} + q_0 \vec{p} + \vec{p} \wedge \vec{q}$$

where $\vec{p} \cdot \vec{q}$ and $\vec{p} \wedge \vec{q}$ denote the inner and outer products of $\vec{p}$ and $\vec{q}$. The product $\mathbf{PQ}$ is linear in $\mathbf{P}$ and $\mathbf{Q}$.

The conjugate $\mathbf{X}^*$ of a quaternion $\mathbf{X} = x_0 + \vec{x}$ is defined by:

$$\mathbf{X}^* = x_0 - \vec{x}.$$

Notice that:

$$\mathbf{X}\mathbf{X}^* = x_0^2 + \|\vec{x}\|^2$$

is a positive or null scalar. $\|\mathbf{X}\| = \sqrt{\mathbf{X}\mathbf{X}^*}$ is the Euclidean length of $\mathbf{X}$ in $\mathbf{R}^4$. $\mathbf{X}$ is called a **unit quaternion** if it satisfies $\mathbf{X}\mathbf{X}^* = 1$.

A rotation around a unit vector $\vec{n}$ by an angle θ can be represented by the following unit quaternion:

$$\mathbf{X}(\vec{n}, \theta) = \cos\frac{\theta}{2} + \vec{n}\sin\frac{\theta}{2}.$$

With this representation, one can verify that:

$$\forall a \in \mathcal{A} : a(\mathbf{q}) = \mathbf{X}_\Theta a(0) \mathbf{X}_\Theta^* + \mathcal{T}$$

where $a(\mathbf{q})$ denotes the coordinate vector of a in $\mathcal{F}_\mathcal{W}$ when $\mathcal{A}$ is at configuration $\mathbf{q} = (\mathcal{T}, \Theta) \in \mathbf{R}^3 \times SO(3)$, and $\mathbf{X}_\Theta$ denotes the unit quaternion representing the orientation Θ in $\mathbf{R}^4$. The expression $\mathbf{X}_\Theta a(0) \mathbf{X}_\Theta^* + \mathcal{T}$ is worth some explanation. In the product $\mathbf{X}_\Theta a(0) \mathbf{X}_\Theta^*$, $a(0)$ is treated as a quaternion having a zero scalar part, i.e. $0 + a(0)$. It is straightforward to verify that the product is also a quaternion having a zero scalar part. The three components of the vector part of $\mathbf{X}_\Theta a(0) \mathbf{X}_\Theta^*$ are added to the components of $\mathcal{T}$, yielding the coordinate vector $a(\mathbf{q})$.

The space of unit quaternions is S^3, the unit 3-sphere in $\mathbf{R}^4$. The same orientation can be represented by two unit quaternions $\mathbf{X}$ and $-\mathbf{X}$. Thus, the application $S^3 \rightarrow SO(3)$ that maps a quaternion to the corresponding orientation in $SO(3)$ is a two-to-one smooth map. Although S^3 is locally diffeomorphic to $SO(3)$, the application $S^3 \rightarrow SO(3)$ is not a diffeomorphism, and S^3 is not an embedding of $SO(3)$ in $\mathbf{R}^4$.

Let $\sim$ be the equivalence relation such that for all $X \in S^3$, $X \sim -X$. The quotient space $S^3/\sim$ with the quotient topology (see Appendix A) is a topological space denoted by P^3 and called **projective 3-space**. It can be regarded as the 3-sphere S^3 with antipodal points identified, or as the space of all lines in $\mathbf{R}^4$ passing through the origin. $SO(3)$ and P^3 are homeomorphic, i.e. related by a continuous bijective map. Hence, they are topologically equivalent.

One can use unit quaternions as follows to construct an atlas of $SO(3)$ made of four charts (U_i, ϕ_i), $i = 0, ..., 3$ [Khatib, 1980] [Canny, 1988]. Let $\mathbf{u}$ be an element of $SO(3)$ represented as a unit quaternion $\mathbf{U} = u_0 + \vec{u} = (u_0, u_1, u_2, u_3)$. Assume that the ith component of $\mathbf{U}$ has the largest magnitude. Then $\mathbf{u} \in U_i$ and $\phi_i(\mathbf{u}) = (\frac{u_j}{|u_i|})_{j=0,...,3\,;\,j\neq i}$. For instance, if $i = 0$, $\phi_0(\mathbf{u}) = (\frac{u_1}{|u_0|}, \frac{u_2}{|u_0|}, \frac{u_3}{|u_0|})$. The four U_i's cover $SO(3)$, but they are closed sets which intersect at their boundaries. One way to obtain open sets covering $SO(3)$ is to define U_i as the subset of all the elements $\mathbf{u} = (u_0, ..., u_3)$ of $SO(3)$ such that for all $j \in [0, 3]$, $j \neq i$, $|u_j| - |u_i| < \varepsilon$ for a given small $\varepsilon > 0$.

The matrix of $SO(3)$ corresponding to the unit quaternion $\mathbf{U} = (u_0, u_1, u_2, u_3)$ is:

$$\begin{pmatrix} 2(u_0^2 + u_1^2) - 1 & 2(u_1u_2 - u_0u_4) & 2(u_1u_3 + u_0u_2) \\ 2(u_1u_2 + u_0u_3) & 2(u_0^2 + u_2^2) - 1 & 2(u_2u_3 - u_0u_1) \\ 2(u_1u_3 - u_0u_2) & 2(u_2u_3 + u_0u_1) & 2(u_0^2 + u_3^2) - 1 \end{pmatrix}.$$

5 Paths and Multiple Connectedness

In this section we define the notion of a path in configuration space. In a Euclidean space, such as $\mathbf{R}^m$, if two paths connect the same initial point to the same final point, either one of them can be "continuously" deformed to the other. In $\mathcal{C}$, this is not true, and $\mathcal{C}$ is said to be multiply connected. This provides evidence that the global topology of $\mathcal{C}$ is

TESLA EXHIBIT 1013, Page 74

different from that of $\mathbf{R}^m$.

5.1 Path in $\mathcal{C}$

The definition of a path in $\mathcal{C}$ is the standard definition of a path in a topological space [Jänich, 1984], i.e. :

DEFINITION 2: *A* **path** *of the robot $\mathcal{A}$ from $\mathbf{q}_{init}$ to $\mathbf{q}_{goal}$ is a continuous map $\tau : [0,1] \rightarrow \mathcal{C}$, with $\tau(0) = \mathbf{q}_{init}$ and $\tau(1) = \mathbf{q}_{goal}$. $\mathbf{q}_{init}$ and $\mathbf{q}_{goal}$ are called the* **initial** *and* **goal** *configurations of the path, respectively.*

This definition directly entails that the set $\{\tau(s) \,/\, s \in [0,1]\}$ is compact. (Indeed, the image of a compact set by a continuous map is a compact set.) This implies that $\tau(s)$ is contained in a bounded domain of $\mathbf{R}^M$.

Now we can formally state the **free-flying object assumption** of the basic motion planning problem as follows: In the absence of obstacles, any path defined as above is feasible. It corresponds to assuming that the motions of $\mathcal{A}$ are not limited by any kinematic or dynamic constraints.

Let us consider any two configurations $\mathbf{q}_{init}$ and $\mathbf{q}_{goal}$ in $\mathcal{C}$. The rigid body transformation that moves $\mathcal{A}$ from $\mathbf{q}_{init}$ to $\mathbf{q}_{goal}$ can be uniquely represented as the composition of a rotation $Rot(\theta)$ by some angle θ around a fixed axis and a translation $Trsl(\delta)$ by some distance δ along a fixed vector. We can write:

$$TR_{\mathbf{q}_{goal}} = Trsl(\delta) \circ Rot(\theta) \circ TR_{\mathbf{q}_{init}},$$

where $TR_{\mathbf{q}_{init}}$ and $TR_{\mathbf{q}_{goal}}$ are the transformations corresponding to $\mathbf{q}_{init}$ and $\mathbf{q}_{goal}$, respectively (see Section 1). The continuous map $\tau : s \in [0,1] \mapsto \tau(s) \in \mathcal{C}$, with

$$TR_{\tau(s)} = Trsl(s\delta) \circ Rot(s\theta) \circ TR_{\mathbf{q}_{init}},$$

is a path from $\mathbf{q}_{init}$ to $\mathbf{q}_{goal}$. Therefore:

PROPOSITION 2: *$\mathcal{C}$ is* **connected**, *i.e. any two configurations are connected by a path.*[6]

[6] In any manifold, connectedness and path-connectedness are two equivalent notions [Bishop and Goldberg, 1980].

This proposition simply states "physical evidence". If $\mathcal{A}$ is a rigid object moving freely in a workspace without obstacles, and $\mathbf{q}_{init}$ and $\mathbf{q}_{goal}$ are configurations of $\mathcal{A}$, then there is a way to move $\mathcal{A}$ from $\mathbf{q}_{init}$ to $\mathbf{q}_{goal}$.

Let τ and τ' be two paths in $\mathcal{C}$ such that $\tau(1) = \tau'(0)$. The path $\tau \bullet \tau'$ defined by:

$$\tau \bullet \tau'(s) = \begin{cases} \tau(2s) & \text{if } 0 \leq s \leq \frac{1}{2}, \\ \tau'(2s-1) & \text{if } \frac{1}{2} \leq s \leq 1, \end{cases}$$

is called the **product** (or composition) of τ by τ'.

The path $\varepsilon_{\mathbf{q}_0}$ such that $\varepsilon_{\mathbf{q}_0}(s) = \mathbf{q}_0$ for all $s \in [0,1]$ is called the **null path** at $\mathbf{q}_0$.

Let τ be a path. The path $\overline{\tau}$, with $\overline{\tau}(s) = \tau(1-s)$ for all $s \in [0,1]$, is called the **inverse path** of τ.

5.2 Homotopy and Fundamental Group

In this subsection we give standard definitions and results from Topology (e.g. see [Massey, 1967] [Jänich, 1984]).

Two paths τ and τ' in a topological space $\mathcal{X}$ are **homotopic** (with endpoints fixed) if there exists a continuous map $h : [0,1] \times [0,1] \to \mathcal{X}$, with $h(s,0) = \tau(s)$ and $h(s,1) = \tau'(s)$ for all $s \in [0,1]$, and $h(0,t) = h(0,0)$ and $h(1,t) = h(1,0)$ for all $t \in [0,1]$. Homotopy determines a relation of equivalence denoted by $\simeq$ among the paths having the same origin and the same final extremity. Obviously, if $\tau_1 \simeq \tau_1'$ and $\tau_2 \simeq \tau_2'$, and $\tau_1(1) = \tau_2(0)$, then $\tau_1 \bullet \tau_2 \simeq \tau_1' \bullet \tau_2'$.

A path-connected space $\mathcal{X}$ is **simply connected** if every two paths τ and τ' such that $\tau(0) = \tau'(0)$ and $\tau(1) = \tau'(1)$ are homotopic; otherwise, it is **multiply connected**. The tool used for characterizing the connectedness of a topological space is the *fundamental group*, which we introduce below.

Let x_0 be an arbitrary point in $\mathcal{X}$. A **loop** based at x_0 is a path λ in $\mathcal{X}$ such that $\lambda(0) = \lambda(1) = x_0$. Let $\Omega(\mathcal{X}, x_0)$ be the set of all the loops in $\mathcal{X}$ based at x_0, and $\pi_1(\mathcal{X}, x_0) = \Omega(\mathcal{X}, x_0)/\simeq$ the set of homotopic classes of loops based at x_0. If λ is a loop of $\Omega(\mathcal{X}, x_0)$, $[\lambda]$ denotes the loop class in $\pi_1(\mathcal{X}, x_0)$ to which it belongs.

The operation $[\lambda] \bullet [\lambda'] \stackrel{\text{def}}{=} [\lambda \bullet \lambda']$ provides $\pi_1(\mathcal{X}, x_0)$ with the structure of a group. The identity element, called the null-homotopic element, is $[\varepsilon_{x_0}]$, where ε_{x_0} is the null path at x_0. The inverse of any $[\lambda] \in \pi_1(\mathcal{X}, x_0)$ is $[\overline{\lambda}]$. If $\mathcal{X}$ is path-connected, the groups $\pi_1(\mathcal{X}, x_0)$ and $\pi_1(\mathcal{X}, x_1)$ are isomorphic for any two points $x_0, x_1 \in \mathcal{X}$. Hence, we will use the notation $\pi_1(\mathcal{X})$ for denoting any of these groups. $\pi_1(\mathcal{X})$ is called the **fundamental group** of $\mathcal{X}$. One can verify that the number of classes of homotopic paths joining any two given points is equal to the cardinality of the fundamental group of the topological space. Thus, a path-connected topological space is simply connected if and only if its fundamental group contains the null-homotopic element as its sole element. It is multiply connected otherwise.

If $\mathcal{X}$ and $\mathcal{Y}$ are two topological spaces, the fundamental group of the product space $\mathcal{X} \times \mathcal{Y}$ is the product of the fundamental groups of the two spaces, i.e. $\pi_1(\mathcal{X} \times \mathcal{Y}) = \pi_1(\mathcal{X}) \times \pi_1(\mathcal{Y})$. In the particular case of $\mathcal{C}$, $\pi_1(\mathcal{C}) = \pi_1(\mathbf{R}^N \times SO(N)) = \pi_1(\mathbf{R}^N) \times \pi_1(SO(N))$.

The fundamental group of $\mathbf{R}^n$ $(n > 0)$ is $\{0\}$, where 0 denotes the null-homotopic element. Hence, every two paths in $\mathbf{R}^n$ with the same origin and the same extremity are homotopic.

In the next two subsections we discuss informally the connectedness of $\mathbf{R}^N \times SO(N)$. A rigorous treatment of this topic can be found in [Massey, 1967].

5.3 Connectedness of $\mathbf{R}^2 \times SO(2)$

We noticed in Subsection 3.3 that $SO(2)$ is diffeomorphic (hence, homeomorphic) to S^1, the unit circle in $\mathbf{R}^2$.

The fundamental group of S^1 is known to be isomorphic to $\mathbf{Z}$, the set of integers with addition as the group operation. This means that there is a countable infinity of homotopic classes of paths between any two configurations $\mathbf{q}_{init}$ and $\mathbf{q}_{goal}$ in $\mathbf{R}^2 \times SO(2)$ (or equivalently, $\mathbf{R}^2 \times S^1$). Each class is uniquely identified by the sign of the net rotation around the circle S^1 and the number of net complete rotations around S^1.

We can graphically illustrate the multiple connectedness of $\mathbf{R}^2 \times S^1$ by considering the space $\mathbf{R} \times S^1$, a cylinder in $\mathbf{R}^3$ shown in Figure 5. No two of the three paths τ_1, τ_2, and τ_3 drawn in the cylinder are homotopic.

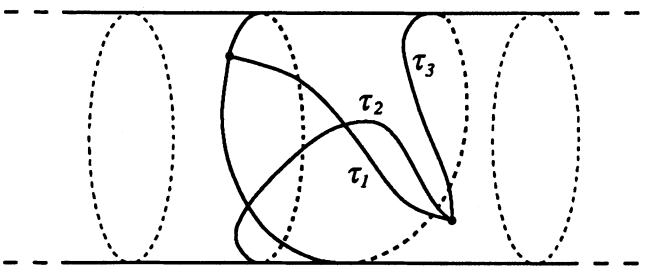


Figure 5. $SO(2)$ (or S^1) is a multiply connected topological space, with $\mathbf{Z}$ as the fundamental group. The three paths τ_1, τ_2, and τ_3 drawn in the infinite cylinder $\mathbf{R} \times S^1$ are not homotopic. Indeed, no two of them wind around the cylinder with the same number of net rotations (with the same sign).

5.4 Connectedness of $\mathbf{R}^3 \times SO(3)$

We noticed in Subsection 4.3 that $SO(3)$ is homeomorphic to the quotient space of the 3-sphere S^3 in $\mathbf{R}^4$ obtained by identifying its antipodal points, i.e. $P^3 = S^3/\sim$.

The fundamental group of $P^n = S^n/\sim$ $(n \geq 2)$ is known to be isomorphic to the quotient set $\mathbf{Z}$ mod 2, i.e. $\mathbf{Z}_2 = \{0, 1\}$. This means that there are two homotopic classes of paths[7] between any two configurations $\mathbf{q}_{init}$ and $\mathbf{q}_{goal}$ in $\mathbf{R}^3 \times P^3$, hence $\mathbf{R}^3 \times SO(3)$.

The null loop class, 0, contains all the loops along which $\mathcal{A}$ undergoes a net rotation of $4k\pi$, with $k \in \mathbf{Z}$, i.e. a net rotation of 0 or an even multiple of 2π. The other loop class, 1, contains all the loops along which $\mathcal{A}$ undergoes a net rotation of $2(2k+1)\pi$, with $k \in \mathbf{Z}$, i.e. a net rotation of an odd multiple of 2π. The fact that any loop along which $\mathcal{A}$ undergoes a net rotation of $4k\pi$ is null-homotopic can be illustrated as follows. Assume that $k = 1$. The rotation during the loop can be expressed as a rotation of 4π around a single axis. Let us decompose this rotation into two successive rotations of 2π each, around this same axis. The second rotation can be homotopically deformed by a continuous change of the orientation of its axis. (This change leaves the initial and final orientations of $\mathcal{A}$ unchanged because the rotation is by an angle of 2π.) Let the change in orientation proceed until the axis of the second rotation

[7] See [Burdick, 1988] for a "robotic-oriented" description of the topology of $SO(3)$.

gets colinear with the axis of the first rotation, with opposite direction. Then the composition of the two rotations produces a net rotation of 0, showing that the original loop is null-homotopic. The same deformation as above cannot be applied to a loop of $2(2k+1)\pi$ net rotation.

Assume that the base configuration of the loop is $\mathbf{q}_0$ and let us select a point p_0 representing this configuration in S^3. A loop in the null loop class corresponds to a continuous curve in S^3 that terminates at p_0, while a loop in the non-null loop class corresponds to a continuous curve in S^3 that terminates at $-p_0$, the antipodal point of p_0. In Topology, S^3 is said to be a twofold covering space of $SO(3)$, and the antipodal identification is the covering map.

Let us consider two paths τ_1 and τ_2, such that $\tau_1(0) = \tau_2(0)$ and $\tau_1(1) = \tau_2(1)$. Let ω_1 be the net rotation that $\mathcal{A}$ undergoes along τ_1 and ω_2 the net rotation along τ_2. If ω_1 and ω_2 differ by 0 or an even multiple of 2π, τ_1 and τ_2 are homotopic; otherwise they are not. Viewed in S^3, if τ_1 and τ_2 are non homotopic, they start at the same point and end at two antipodal points.

6 Directions of Motion

In this section we introduce the necessary notions that make it possible to talk about directions in configuration space. We define the tangent vector to a differentiable curve contained in $\mathcal{C}$. The space spanned by all the possible tangent vectors at a given configuration is a vector space, called the *tangent space.* One such space can be attached with each configuration $\mathbf{q}$ in $\mathcal{C}$.

6.1 Tangent Space

Consider a curve γ:

$$s \in \mathbf{R} \mapsto \gamma(s) \in \mathcal{C}.$$

Let (U, ϕ) be a chart in $\mathcal{C}$ such that $\gamma(s_0) \in U$. We denote by $x_i(\mathbf{q})$, $i = 1, ..., m$, the coordinates of any configuration $\mathbf{q} \in U$ given by ϕ. γ is **differentiable** at $s = s_0$ if and only if for every $i \in [1, m]$, $x_i(\gamma(s))$ is a differentiable function of s at s_0. Notice that if this property is verified for a chart (U, ϕ), then it also holds for any other chart (V, ψ) in the atlas of $\mathcal{C}$ such that $\gamma(s_0) \in V$, since (U, ϕ) and (V, ψ) are C^∞-related. A curve

is differentiable[8] if it is differentiable at all the values of s in its definition domain ($\mathbf{R}$ or an open interval of $\mathbf{R}$). The notion of differentiability is extended in the usual fashion to curves γ defined on a closed interval by using the right and left derivatives at the extremities of the interval.

Let us consider the embedding $\mathcal{M}_\mathcal{C}$ of $\mathcal{C}$ in $\mathbf{R}^M$ that we defined in Section 2. At every point $\mathbf{u}$ in $\mathcal{M}_\mathcal{C}$, the $M - m$ independent gradient vectors $\vec{\nabla}h_1, ..., \vec{\nabla}h_{M-m}$ (see Subsection 3.1) span a vector subspace of $\mathbf{R}^M$ of dimension $M - m$. The orthogonal complement to this subspace in $\mathbf{R}^M$ is a subspace of dimension m, called the tangent space of $\mathcal{M}_\mathcal{C}$ at $\mathbf{u}$. For a moment, let us denote this space by $T_\mathbf{u}$.

Let u be the map embedding $\mathcal{C}$ in $\mathbf{R}^M$, with:

$$u : \mathbf{q} \in \mathcal{C} \mapsto u(\mathbf{q}) = (u_1(\mathbf{q}), ..., u_M(\mathbf{q})) \in \mathcal{M}_\mathcal{C}.$$

Let γ be a differentiable curve in $\mathcal{C}$. The representation of γ in $\mathcal{M}_\mathcal{C}$, γ', is defined by:

$$s \in \mathbf{R} \mapsto \gamma'(s) = (u_1(\gamma(s)), ..., u_M(\gamma(s))) \in \mathcal{M}_\mathcal{C}.$$

It is a differentiable function of s and the M-vector

$$\frac{d\vec{\gamma}'(s)}{ds}_{|s=s_0} = \left(\frac{du_i(\gamma(s))}{ds}_{|s=s_0}\right)_{i=1,...,M}$$

is the tangent vector to γ' at $s = s_0$. This vector belongs to $T_{\mathbf{u}_0}$, where $\mathbf{u}_0 = \gamma'(s_0) = u(\gamma(s_0))$. Thus, the tangent space $T_{\mathbf{u}_0}$ can be regarded as the space spanned by the tangent vectors to all the differentiable curves at $\mathbf{u}_0 \in \mathcal{M}_\mathcal{C}$.

Tangent space and tangent vectors can be defined independently of the embedding of $\mathcal{C}$ into $\mathbf{R}^M$. Two differentiable curves γ_1 and γ_2 in $\mathcal{C}$ are tangent at $s = s_0$ if $\gamma_1(s_0) = \gamma_2(s_0) = \mathbf{q}_0$ and, if in some chart (U, ϕ) in the atlas of $\mathcal{C}$ such that $\mathbf{q}_0 \in U$, we have:

$$\forall i \in [1, m] \; : \; \lim_{s \to s_0} \frac{x_i(\gamma_1(s)) - x_i(\gamma_2(s))}{s - s_0} = 0,$$

i.e. $(dx_i(\gamma_1(s))/ds)_{|s=s_0} = (dx_i(\gamma_2(s))/ds)_{|s=s_0}$, for $i = 1, ..., m$. Again, since overlapping charts are C^∞-related, this relationship is independent

[8] We say that the curve is smooth when the functions $x_i(\gamma(s))$ are smooth, i.e. have derivatives of all orders.

TESLA EXHIBIT 1013, Page 80

of the chart at $\mathbf{q}_0$ in the atlas. It defines an equivalence class of paths passing through $\mathbf{q}_0$ in $\mathcal{C}$, which is characterized by the m numbers:

$$\xi_i = \frac{dx_i(\gamma_1(s))}{ds}_{|s=s_0} = \frac{dx_i(\gamma_2(s))}{ds}_{|s=s_0}.$$

Such an equivalent class is called a **tangent vector** of $\mathcal{C}$ at $\mathbf{q}_0$.

Given two tangent vectors $\vec{\xi}$ and $\vec{\zeta}$ at $\mathbf{q}$, it is easy to construct curves γ and γ' passing through $\mathbf{q}$, such that γ and γ' admit $\lambda\vec{\xi}$ and $\vec{\xi}+\vec{\zeta}$ as their respective tangent vectors at $\mathbf{q}$. Hence, the set of tangent vectors of $\mathcal{C}$ at any $\mathbf{q} \in \mathcal{C}$ forms an m-dimensional vector space. This space is called the **tangent space** of $\mathcal{C}$ at $\mathbf{q}$. From now on, we will denote it by $T_{\mathbf{q}}(\mathcal{C})$.

We can define an inner product in a tangent space. Such an inner product induces a norm and makes it possible to compare the orientations of two differentiable curves at an intersection point. A manifold with an inner product on every tangent space is called a **Riemannian manifold.** The norm associated with this inner product is called the **Riemannian metric** [Arnold, 1978].

Let $\gamma : t \in [0,T] \mapsto \gamma(t) \in \mathcal{C}$, where t denotes time, be a differentiable curve representing a motion of $\mathcal{A}$. At any time $t \geq 0$, the tangent vector to γ at the current configuration represents the velocity of $\mathcal{A}$. Thus, the tangent space is also called *velocity space.*

It is sometimes convenient to blend the tangent spaces at the various points of $\mathcal{C}$ into a single structure. The union of the tangent spaces at the various points of $\mathcal{C}$, $T(\mathcal{C}) = \bigcup_{\mathbf{q}\in\mathcal{C}} T_{\mathbf{q}}(\mathcal{C})$, is called the **tangent bundle** of $\mathcal{C}$. It is a smooth manifold of dimension $2m$ [Spivak, 1979]. Since a pair (configuration, velocity) is called a phase in Physics and a state in Control Theory, the tangent bundle of $\mathcal{C}$ is also named *phase space* and *state space.*

6.2 Basis Induced by a Chart

Let (U,ϕ) be a chart and X be the image of U in $\mathbf{R}^m$ by ϕ. A basis of $\mathbf{R}^m$ and the chart (U,ϕ) **induce** an ordered basis[9] $(\vec{t}_1,...,\vec{t}_m)$ in $T_{\mathbf{q}}(\mathcal{C})$, which is defined as follows. Each vector $\vec{t}_i$ is the tangent vector at $\mathbf{q}$ to

[9]Ordering the vectors $\vec{t}_i$, $i = 1,...,m$, makes sense because $\mathbf{R}^N \times SO(N)$ is orientable [Guillemin and Pollack, 1974] [Spivak, 1979].

any curve γ that maps in $\mathbf{R}^m$ to a curve $\phi \circ \gamma$ tangent at $\phi(\mathbf{q})$ to the ith vector of the basis of $\mathbf{R}^m$. The m numbers $\xi_i = \frac{dx_i(\gamma(s))}{ds}|_{s=s_0}$ are the **components** of the tangent vector of γ at $\mathbf{q}_0 = \gamma(s_0)$ in this basis.

Consider two charts (U, ϕ) and (U', ϕ'), with $\mathbf{q} \in U \cap U'$. Let β and β' be the bases induced by these charts in $T_{\mathbf{q}}(\mathcal{C})$ and $f = \phi' \circ \phi^{-1}$, with:

$$f : (x_1, ..., x_m) \in \mathbf{R}^m \mapsto (f_1(x_1, ..., x_m), ..., f_m(x_1, ..., x_m)) \in \mathbf{R}^m.$$

The **Jacobian matrix** of the map f, denoted by $J_f(x_1, ..., x_m)$, is the $m \times m$ matrix of the first derivatives of f, i.e. :

$$J_f(x_1, ..., x_m) = \begin{pmatrix} \frac{\partial f_1}{\partial x_1}(x_1, ..., x_m) & \cdots & \frac{\partial f_1}{\partial x_m}(x_1, ..., x_m) \\ \cdot & & \cdot \\ \cdot & & \cdot \\ \cdot & & \cdot \\ \frac{\partial f_m}{\partial x_1}(x_1, ..., x_m) & \cdots & \frac{\partial f_m}{\partial x_m}(x_1, ..., x_m) \end{pmatrix}.$$

Let $\vec{t}$ be a vector of $T_{\mathbf{q}}(\mathcal{C})$. We denote by $[\vec{t}]_\beta$ and $[\vec{t}]_{\beta'}$ the m-vectors of $\vec{t}$'s components in β and β'. We have:

$$[\vec{t}]_{\beta'} = J_f(x_1(\mathbf{q}), ..., x_m(\mathbf{q}))\, [\vec{t}]_\beta.$$

The basis induced in $T_{\mathbf{q}}(\mathcal{C})$ by the chart (U, ϕ) makes it possible to use the same copy of $\mathbf{R}^m$ for representing both the set $\phi(U)$ and the tangent spaces of $\mathcal{C}$ at all the configurations in U. However, a single vector of $\mathbf{R}^m$ does not determine a unique direction of motion relative to $\mathcal{F}_\mathcal{A}$. This direction depends on the current configuration of $\mathcal{A}$.

6.3 Principal Axis Basis

Consider the case where $N = 3$. It is often convenient to express the velocity of $\mathcal{A}$ as a six-dimensional vector $(\dot{x}_\mathcal{A}, \dot{y}_\mathcal{A}, \dot{z}_\mathcal{A}, \dot{\alpha}, \dot{\beta}, \dot{\gamma})$, where $\dot{x}_\mathcal{A}$, $\dot{y}_\mathcal{A}$, and $\dot{z}_\mathcal{A}$ are the linear velocities along the axes of $\mathcal{F}_\mathcal{A}$, while $\dot{\alpha}$, $\dot{\beta}$, and $\dot{\gamma}$ are the angular velocities around these axes. Pure instantaneous translations along each of the axes of $\mathcal{F}_\mathcal{A}$ and pure instantaneous rotations around each of these axes determine a vector basis in every tangent space, which we call the **principal axis basis.**[10]

[10]This name derives from the fact that $\mathcal{A}$'s principal axes of inertia are a natural choice for $\mathcal{F}_\mathcal{A}$'s axes.

TESLA EXHIBIT 1013, Page 82

Similarly, in the two-dimensional case, the principal axis basis consists of three vectors corresponding to two instantaneous translations along the axes of $\mathcal{F}_\mathcal{A}$ and an instantaneous rotation around the origin $O_\mathcal{A}$.

As we will verify below, when $N = 3$, the principal axis basis is not induced by any chart in $\mathcal{C}$. However, its relationship with a basis induced by a chart can be established. For instance, let $(\dot{x}_\mathcal{A}, \dot{y}_\mathcal{A}, \dot{z}_\mathcal{A}, \dot{\alpha}, \dot{\beta}, \dot{\gamma})$ be a tangent vector represented in the principal axis basis and $(\dot{x}_\mathcal{W}, \dot{y}_\mathcal{W}, \dot{z}_\mathcal{W}, \dot{\phi}, \dot{\theta}, \dot{\psi})$ be the same vector represented in the chart induced by the Eulerian chart defined in Subsection 4.2. We have:

$$\begin{pmatrix} \dot{x}_\mathcal{A} \\ \dot{y}_\mathcal{A} \\ \dot{z}_\mathcal{A} \end{pmatrix} = \mathbf{E}^T_{\phi\theta\psi} \begin{pmatrix} \dot{x}_\mathcal{W} \\ \dot{y}_\mathcal{W} \\ \dot{z}_\mathcal{W} \end{pmatrix}$$

where $\mathbf{E}^T_{\phi\theta\psi}$ denotes the transpose of the $SO(3)$ matrix[11] corresponding to the Euler angles (ϕ, θ, ψ), and:

$$\begin{pmatrix} \dot{\alpha} \\ \dot{\beta} \\ \dot{\gamma} \end{pmatrix} = \begin{pmatrix} -\sin\theta\cos\psi & \sin\psi & 0 \\ \sin\theta\sin\psi & \cos\psi & 0 \\ \cos\theta & 0 & 1 \end{pmatrix} \begin{pmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{pmatrix}.$$

The determinant of the second system is $-\sin\theta$, which is non-zero for $0 < \theta < \pi$. One can also verify that the second system cannot be integrated, i.e. $\dot{\alpha}$, $\dot{\beta}$ and $\dot{\gamma}$ are not the differentials of functions of the Eulerian angles. Indeed, from the third equation, it follows that $\frac{\partial\gamma}{\partial\phi} = \cos\theta$ and $\frac{\partial\gamma}{\partial\theta} = 0$. Hence $\frac{\partial}{\partial\theta}\frac{\partial\gamma}{\partial\phi} \neq \frac{\partial}{\partial\phi}\frac{\partial\gamma}{\partial\theta}$. If the principal axis basis was induced by a chart in $\mathcal{C}$, $\dot{\alpha}$, $\dot{\beta}$ and $\dot{\gamma}$ would be the differentials of coordinates α, β and γ, which could be expressed in terms of ϕ, θ and ψ.

When the principal axis basis is used, a vector of $\mathbf{R}^m$, with $\mathbf{R}^m$ being considered as a representation of the tangent spaces of $\mathcal{C}$, determines a unique direction of motion relative to $\mathcal{F}_\mathcal{A}$, independent of the configuration of $\mathcal{A}$.

6.4 Remark on the Differentiability of Paths

Let us consider a path $\tau : s \in [0,1] \mapsto \tau(s) \in \mathcal{C}$. A bijective function $f : s \in [0,1] \mapsto f(s) \in [0,1]$, with $f(0) = 0$ and $f(1) = 1$, is called a

[11] The inverse of a matrix in $SO(N)$ is equal to the transpose of this matrix.

reparameterization of τ. The path $\tau' : s \in [0,1] \mapsto \tau'(s) = \tau(f(s)) \in \mathcal{C}$ produces the same geometric curve segment as τ, but the rate at which this segment is traced out when s varies from 0 to 1 is different.

Assume that τ is differentiable at every $s \in [0,1]$, except possibly at a finite number of values of s. The non-differentiability of τ at some points may simply be an effect of the parameterization of the path. For example, if τ is not differentiable at s, but if its left and right tangent vectors at s:

$$\lim_{\varepsilon \to 0} \frac{d\vec{\tau}}{ds}_{|s-\varepsilon} \quad \text{and} \quad \lim_{\varepsilon \to 0} \frac{d\vec{\tau}}{ds}_{|s+\varepsilon}$$

exist, are non-zero, and have the same orientation, i.e. :

$$\lim_{\varepsilon \to 0} \frac{d\vec{\tau}}{ds}_{|s-\varepsilon} = \alpha \lim_{\varepsilon \to 0} \frac{d\vec{\tau}}{ds}_{|s+\varepsilon} \quad \text{with} \quad \alpha > 0,$$

then we can reparameterize τ into a path τ' that produces the same curve segment as τ and is differentiable at s.

If τ is differentiable everywhere except possibly at a finite number of points, it has a finite length (using, for instance, the Euclidean metric in $\mathbf{R}^M$). Let $l : [0,1] \to \mathbf{R}$ be the function that measures the length between $\tau(0)$ and $\tau(s)$ along τ. $L = \tau(1)$ is the length of the curve segment representing τ. The function $l(s)/L$ (normalized length) is a reparameterization of τ.

Let τ be a path whose parameter denotes the normalized length. If τ is differentiable at s, its tangent vector at s is non-zero. If τ is non-differentiable at s, neither the left nor the right tangent vector at s are zero and they do not have the same orientation. The direction of the tangent vector at $\tau(s)$, if τ is differentiable at s, determines the direction of the velocity of $\mathcal{A}$ for any execution of the path. The velocity of a physical object $\mathcal{A}$ cannot change instantaneously without consuming infinite energy. Therefore, the points where τ is not differentiable determine configurations where $\mathcal{A}$ has to stop.

A path τ is said to be of **class** C^1 if the reparameterized path using the normalized length is differentiable everywhere. It is said to be **piecewise of class** C^1 if the reparameterized path is differentiable everywhere except possibly at a finite number of values of s.

TESLA EXHIBIT 1013, Page 84

A path planner should only generate paths of class C^1 or piecewise of class C^1. Indeed, although we assume no dynamic constraint in the basic motion planning problem, there is little interest in generating paths which we know in advance to be infeasible.

All the path planning methods presented later directly generate piecewise differentiable paths. There is nothing surprising in that. On the one hand, non-piecewise differentiable paths are not the easiest to build. On the other hand, they are not needed. Indeed, every path τ in $\mathcal{C}$ is homotopic to a piecewise smooth path, which can be made arbitrarily close to it — i.e. for every $\varepsilon > 0$, there exists a piecewise smooth path τ' such that:

$$\forall s \in [0,1]: \; d(\tau(s), \tau'(s)) < \varepsilon,$$

where d is the metric in use in $\mathcal{C}$ (see Section 7). This may be seen by covering τ with a finite number of open subsets U_i, $i = 1, 2, ...$, of $\mathcal{C}$ (remember that τ is compact). Using charts of $\mathcal{C}$, each U_i can be mapped to an open subset X_i of $\mathbf{R}^m$. Thus, τ is homotopic to a piecewise smooth path τ', defined as the inverse image of a finite sequence of straight segments contained in the X_i's. The path τ' can be made as close as one wishes to τ by taking each of the U_i's small enough. More generally, using Weierstrass' Approximation Theorem, one can show that any path τ in $\mathcal{C}$ is homotopic to a smooth path, which can be made as close as one wishes to τ. Most path planning methods, however, do not produce smooth paths directly.

7 Metrics

We mentioned earlier that several metrics yield the same topology of $\mathcal{C}$ as the one previously defined. In this section we give several such metrics. Metrics play no role in the definition of the basic motion planning problem. However, some planning methods use distances to the obstacles and to the goal configuration as heuristic functions in order to guide the search process.

One straightforward way to define a distance function in $\mathcal{C}$ is to let $\mathcal{C}$ inherit the Euclidean distance of the ambient space $\mathbf{R}^M$, i.e. if $\mathbf{q}_1$ and $\mathbf{q}_2$ are two configurations mapping to $\mathbf{u}_1$ and $\mathbf{u}_2$ in $\mathcal{M}_\mathcal{C} \subset \mathbf{R}^M$, then:

$$d(\mathbf{q}_1, \mathbf{q}_2) = \|\mathbf{u}_1 - \mathbf{u}_2\|.$$

A second way to define a distance in $\mathcal{C}$ is by making use of the Euclidean distance in the workspace $\mathcal{W}$. For example, the two functions $d : \mathcal{C} \times \mathcal{C} \rightarrow \mathbf{R}$ specified by:

$$d(\mathbf{q}_1, \mathbf{q}_2) = \max_{a \in \mathcal{A}} \|a(\mathbf{q}_1) - a(\mathbf{q}_2)\|$$

$$d(\mathbf{q}_1, \mathbf{q}_2) = \sum_{k \in [1,N]} \|a_k(\mathbf{q}_1) - a_k(\mathbf{q}_2)\|$$

where a_k, $k = 1, ..., N$, are distinct points in $\mathcal{A}$ (if $N = 3$, the points should not be aligned), are distances in $\mathcal{C}$. (The first of these two functions is a distance because $\mathcal{A}$ is compact.)

Another example of a metric in $\mathcal{C}$ whose definition makes use of the Euclidean distance in the workspace is known as the Hausdorff distance [Grünbaum, 1967]. It requires $\mathcal{A}$ to be compact and is defined as follows:

$$d(\mathbf{q}_1, \mathbf{q}_2) = \max\{d_h(\mathcal{A}(\mathbf{q}_1), \mathcal{A}(\mathbf{q}_2)), d_h(\mathcal{A}(\mathbf{q}_2), \mathcal{A}(\mathbf{q}_1))\}$$

with:

$$d_h(U, V) = \max_{u \in U} \min_{v \in V} \|u - v\|.$$

(Notice that in general: $d_h(U, V) \neq d_h(V, U)$.) For any compact subset S of $\mathbf{R}^N$ and $\varepsilon > 0$, let $S_\varepsilon = \{x \in \mathbf{R}^N \ / \ \exists y \in S : \|x - y\| \leq \varepsilon\}$. Then d_h can also be defined by: $d_h(U, V) = \min\{\varepsilon > 0 \ / \ U \subseteq V_\varepsilon \text{ and } V \subseteq U_\varepsilon\}$.

One can easily verify that all the metrics proposed above induce the same topology of $\mathcal{C}$.

A third way to define a metric in $\mathcal{C}$ is to make $\mathcal{C}$ into a Riemannian manifold by defining an inner product on every tangent space $T_{\mathbf{q}}(\mathcal{C})$. (We denote by $\|\vec{\xi}\|$ the norm of a vector $\vec{\xi} \in T_{\mathbf{q}}(\mathcal{C})$ associated with this inner product.) Let $\gamma : s \in [0, 1] \mapsto \gamma(s) \in \mathcal{C}$ be a smooth curve in $\mathcal{C}$ and $\frac{d\vec{\gamma}(s)}{ds}$ be the tangent vector of γ at s. The integral:

$$L(\gamma) = \int_0^1 \|\frac{d\vec{\gamma}(s)}{ds}\| ds$$

is well-defined and is called the *length* of γ induced by the Riemannian metric. Since $\mathcal{C}$ is connected:

$$d(\mathbf{q}_1, \mathbf{q}_2) = \inf_{\gamma} \{L(\gamma)\}$$

TESLA EXHIBIT 1013, Page 86

is a distance in $\mathcal{C}$ [Spivak, 1979].

One can show that the above distance induces the same topology in $\mathcal{C}$ as the previous metrics (see [Spivak, 1979], pp. 428–429). $\mathcal{C}$ is also "geodesically complete" for this metric (see [Spivak, 1979], pp. 462–464). This means that between any two configurations $\mathbf{q}_1$ and $\mathbf{q}_2$, there exists at least one curve, called the *geodesic of minimal length*, whose length is equal to $d(\mathbf{q}_1, \mathbf{q}_2)$. This curve may not be unique, e.g. there are two curves of minimal length between any two antipodal points of S^1.

8 Particular Cases

So far, we have assumed that the robot $\mathcal{A}$ has no symmetry. Removing this restriction yields particular cases considered below. Some of these cases are of a particular interest because they result in configuration spaces of lower dimensions than $\mathbf{R}^N \times SO(N)$, or even in Euclidean configuration spaces, in which path planning concepts and methods are easier to introduce.

If $\mathcal{A}$ is an N-ball[12] with no mark on it, we cannot distinguish the configurations that only differ by a rotation around the center of the ball. Then, by choosing $O_\mathcal{A}$ at the center of the ball, the configuration space can be represented as the N-dimensional Euclidean space $\mathbf{R}^N$.

If $\mathcal{A}$ is a cylinder of finite length[13] in $\mathbf{R}^3$, we cannot distinguish between two configurations that only differ by a rotation around the axis of the cylinder. Then, choosing $O_\mathcal{A}$ on $\mathcal{A}$'s axis and one of $\mathcal{F}_\mathcal{A}$'s axes along $\mathcal{A}$'s axis leads to representing $\mathcal{C}$ as $\mathbf{R}^3 \times S^2$, where S^2 denotes the unit sphere in $\mathbf{R}^3$. Thus, in this case, $\mathcal{C}$ is a five-dimensional manifold. S^2 can be parameterized using the stereographic projection or the spherical angles.

Non-revolute axial symmetries in $\mathcal{A}$ do not reduce the dimension of $\mathcal{C}$. However, they transform the configuration space into a quotient space of $\mathbf{R}^N \times SO(N)$. For instance, a square $\mathcal{A}$ in $\mathbf{R}^2$ is invariant under a rotation by $\pi/2$ around its center. By choosing $O_\mathcal{A}$ at the center of the square, we can represent $\mathcal{C}$ as $\mathbf{R}^2 \times [SO(2)/\{\theta \sim \theta + k\pi/2\}]$, where

[12] This includes the case where $\mathcal{A}$ is a single point in $\mathbf{R}^N$.
[13] This includes the case where $\mathcal{A}$ is a line segment in $\mathbf{R}^3$.

$SO(2)/\{\theta \sim \theta + k\pi/2\}$ is the group of planar rotations modulo $\pi/2$. We will not consider this particular case further in the book.

An interesting case occurs when $\mathcal{A}$ is a dimensioned object which can translate but not rotate. Then $\mathcal{C}$ can be represented as $\mathbf{R}^N$, with a configuration $\mathbf{q}$ being defined as $O_\mathcal{A}$'s coordinates in $\mathcal{F}_\mathcal{W}$. If any path in $\mathcal{C} = \mathbf{R}^N$ is possible, we say that $\mathcal{A}$ is a **free-translating object**. This situation is similar to that obtained when $\mathcal{A}$ is a free-flying N-ball.

Whenever $\mathcal{C} = \mathbf{R}^N$, we can use a single copy of $\mathbf{R}^N$ to represent both $\mathcal{C}$ and all the tangent spaces.

9 Obstacles

Let $\mathcal{W}$ now contain fixed obstacles $\mathcal{B}_i$, $i = 1, 2, ..., q$, which we represent as closed, but not necessarily bounded, regions of $\mathbf{R}^N$. The $\mathcal{B}_i$'s denote both the "physical" obstacles and the subsets of $\mathbf{R}^N$ that represent them.

Given a region $\mathcal{S}$ in a topological space $\mathcal{X}$, we denote the interior, boundary, and closure of $\mathcal{S}$, by $int(\mathcal{S})$, $\partial\mathcal{S}$, and $cl(\mathcal{S})$, respectively, and the complement of $\mathcal{S}$ in $\mathcal{X}$ by $\mathcal{X}\backslash\mathcal{S}$. By definition, $\partial\mathcal{S} = cl(\mathcal{S})\backslash int(\mathcal{S})$.

9.1 C-Obstacles

DEFINITION 3: *The obstacle $\mathcal{B}_i$ in $\mathcal{W}$ maps in $\mathcal{C}$ to the region $\mathcal{CB}_i = \{\mathbf{q} \in \mathcal{C} \,/\, \mathcal{A}(\mathbf{q}) \cap \mathcal{B}_i \neq \emptyset\}$. $\mathcal{CB}_i$ is called a* **C-obstacle.**

The union of all the C-obstacles in $\mathcal{C}$ is called the **C-obstacle region.** We will investigate the representation of C-obstacles in detail in Chapter 3, when both $\mathcal{A}$ and $\mathcal{B}$ are represented as semi-algebraic subsets of $\mathbf{R}^N$. Below, we illustrate the notion of a C-obstacle with very simple examples.

Examples:

- Let $\mathcal{A}$ be a disc, with $O_\mathcal{A}$ at the center of the disc, and $\mathcal{B}$ be a polygon in $\mathcal{W} = \mathbf{R}^2$. Then $\mathcal{C} = \mathbf{R}^2$. $\mathcal{CB}$ is obtained by "growing" $\mathcal{B}$ isotropically by the radius of $\mathcal{A}$ (see Figure 6). The boundary of $\mathcal{CB}$ is the curve followed by $O_\mathcal{A}$ when $\mathcal{A}$ "rolls" over the boundary of $\mathcal{B}$.

- Figure 7 illustrates the case where both $\mathcal{A}$ and $\mathcal{B}$ are convex polygons in $\mathbf{R}^2$, $\mathcal{A}$ being able to translate freely at fixed orientation. It shows

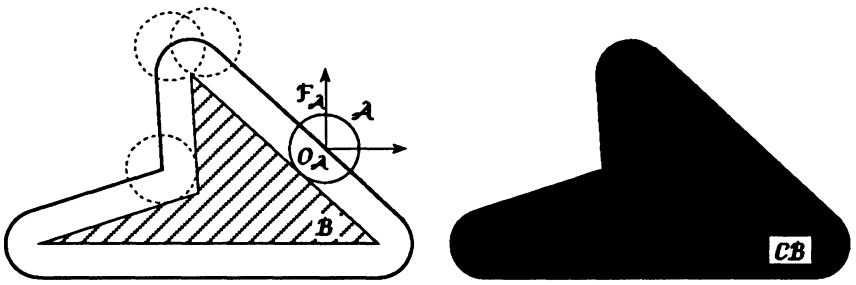


Figure 6. $\mathcal{A}$ is a disc in $\mathbf{R}^2$ and the origin of $\mathcal{F}_\mathcal{A}$ is chosen at the center of the disc. $\mathcal{B}$ is a polygon. The C-obstacle $\mathcal{CB}$ is obtained by growing $\mathcal{B}$ isotropically by the radius of $\mathcal{A}$.

the C-obstacles in $\mathbf{R}^2$ obtained for two different fixed orientations of $\mathcal{A}$. The boundary of each C-obstacle is the curve followed by $O_\mathcal{A}$ when $\mathcal{A}$ slides in contact with $\mathcal{B}$'s boundary, without overlapping of $\mathcal{A}$'s and $\mathcal{B}$'s interiors. Each C-obstacle is a convex polygon.

- Figure 8 displays a representation of the C-obstacle $\mathcal{CB}$ when $\mathcal{A}$ and $\mathcal{B}$ are the same convex polygons as in the previous example, but $\mathcal{A}$ can both translate and rotate. The representation is given by parameterizing each configuration $\mathbf{q}$ by $(x, y, \theta) \in \mathbf{R}^2 \times [0, 2\pi]$. Each cross-section through the C-obstacle, perpendicular to the θ-axis, is the two-dimensional C-obstacle at the corresponding fixed orientation of $\mathcal{A}$. ∎

Let us denote by $\mathcal{CB}_\Theta$ the cross-section of the representation of $\mathcal{CB}$ in $\mathbf{R}^N \times SO(N)$ at a fixed orientation $\Theta \in SO(N)$ of $\mathcal{A}$, by $\mathcal{A}_\Theta$ the object $\mathcal{A}$ at that orientation, and by $\mathcal{A}_\Theta(\mathcal{T})$ the region of $\mathbf{R}^N$ occupied by $\mathcal{A}_\Theta$ at position $\mathcal{T} \in \mathbf{R}^N$. The following proposition relates the shape $\mathcal{CB}_\Theta$ to the shapes of $\mathcal{A}$ and $\mathcal{B}$ [Lozano-Pérez, 1981]. It is illustrated by the above three examples.

PROPOSITION 3: *Posing:*

$$\mathcal{B} \ominus \mathcal{A}_\Theta(0) = \{x \in \mathcal{W} \,/\, \exists b \in \mathcal{B}, \exists a_0 \in \mathcal{A}_\Theta(0) : x = b - a_0\},$$

we have: $\mathcal{CB}_\Theta = \mathcal{B} \ominus \mathcal{A}_\Theta(0)$.

($\ominus$ denotes the Minkowski operator for affine set difference.)

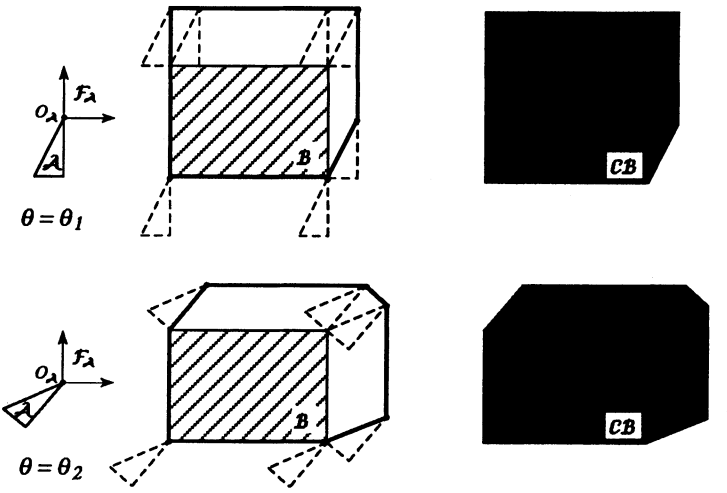


Figure 7. $\mathcal{A}$ and $\mathcal{B}$ are convex polygons. $\mathcal{A}$ can translate freely, but cannot rotate. Therefore, $\mathcal{C} = \mathbf{R}^2$. The figure shows two C-obstacles corresponding to two different fixed orientations of $\mathcal{A}$.

Proof:

1. $\forall a \in \mathcal{A}_\Theta(\mathcal{T})$: $a = a_0 + \mathcal{T}$, with $a_0 \in \mathcal{A}_\Theta(0)$.
   $\forall b \in \mathcal{A}_\Theta(\mathcal{T}) \cap \mathcal{B}$: $\exists a_0 \in \mathcal{A}_\Theta(0)$ such that $b = a_0 + \mathcal{T}$.
   $\forall \mathcal{T} \in \mathcal{CB}_\Theta$: $\mathcal{A}_\Theta(\mathcal{T}) \cap \mathcal{B} \neq \emptyset$.
   Hence, $\mathcal{T} \in \mathcal{B} \ominus \mathcal{A}_\Theta(0)$ and $\mathcal{CB}_\Theta \subseteq \mathcal{B} \ominus \mathcal{A}_\Theta(0)$.

2. $\forall a_0 \in \mathcal{A}_\Theta(0)$: $a_0 + \mathcal{T} \in \mathcal{A}_\Theta(\mathcal{T})$.
   $\forall \mathcal{T} \in \mathcal{B} \ominus \mathcal{A}_\Theta(0)$: $\exists b \in \mathcal{B}$ and $\exists a_0 \in \mathcal{A}_\Theta(0)$ such that $\mathcal{T} = b - a_0$.
   Hence, this same b is an element of $\mathcal{A}_\Theta(\mathcal{T}) \cap \mathcal{B}$, and $\mathcal{B} \ominus \mathcal{A}_\Theta(0) \subseteq \mathcal{CB}_\Theta$.

Therefore: $\mathcal{CB}_\Theta = \mathcal{B} \ominus \mathcal{A}_\Theta(0)$. ∎

Since the Minkowski sum of two convex sets is convex, $\mathcal{CB}_\Theta$ is convex whenever both $\mathcal{A}$ and $\mathcal{B}$ are convex.

9.2 Compactness and Connectedness

We show below that:

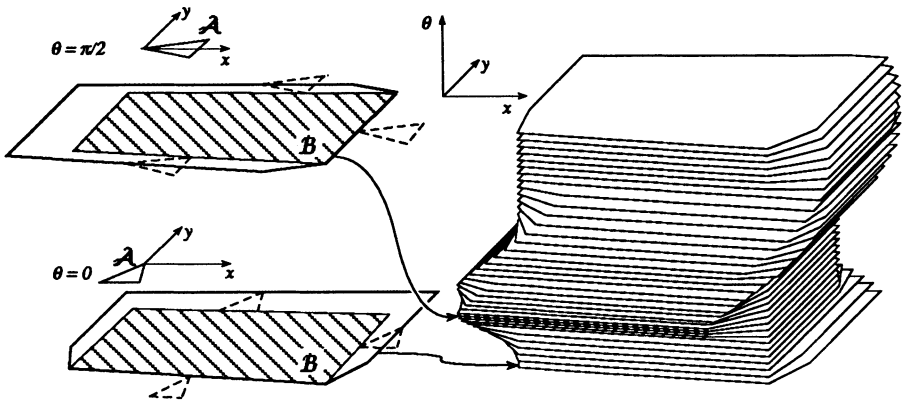


Figure 8. $\mathcal{A}$ and $\mathcal{B}$ are convex polygons. $\mathcal{A}$ is a free-flying object. $\mathcal{C}$ is represented by parameterizing each configuration $\mathbf{q}$ by $(x, y, \theta) \in \mathbf{R}^2 \times [0, 2\pi]$. The represented C-obstacle is a volume bounded by patches of ruled surfaces (see Chapter 3).

- $\mathcal{CB}$ is compact if $\mathcal{B}$ is compact (we already required that $\mathcal{A}$ be compact);
- $\mathcal{CB}$ is closed;
- $\mathcal{CB}$ is connected if both $\mathcal{A}$ and $\mathcal{B}$ are connected.

PROPOSITION 4: *If $\mathcal{B}$ is compact, so is $\mathcal{CB}$.*

Proof: We first show that $\mathcal{CB}$ is closed. Then, embedding $\mathcal{C}$ into $\mathbf{R}^M$, we show that $\mathcal{CB}$ is a bounded subset of $\mathbf{R}^M$. As a closed bounded subset of $\mathbf{R}^M$, $\mathcal{CB}$ is compact.

(1) If for any sequence $\{\mathbf{q}_i\}$, $i = 1, 2, ...,$ of configurations in $\mathcal{CB}$ such that $\lim_{i\to\infty} \mathbf{q}_i = \mathbf{q}$, we can prove that $\mathbf{q}$ is in $\mathcal{CB}$, then $\mathcal{CB}$ is closed. Indeed, the configuration $\mathbf{q}$ defined as the limit of a sequence of configurations in $\mathcal{CB}$ is called a *limit point* of $\mathcal{CB}$, and a set that contains all its limit points is a closed set.

Given $\{\mathbf{q}_i\}$, we can define two sequences of points, $\{a_i\}$ in $\mathcal{A}$ and $\{b_i\}$ in $\mathcal{B}$, such that for every i, $a_i(\mathbf{q}_i)$ and b_i coincide in $\mathcal{W}$. These two sequences exist since $\mathbf{q}_i \in \mathcal{CB}$ for all i. Since $\mathcal{A}$ and $\mathcal{B}$ are both compact subsets of the Euclidean space $\mathbf{R}^N$, they are both compact metric sets, and their

cross-product is also a compact metric set. A compact metric set is also sequentially compact, which means that every sequence in the set has a convergent subsequence [Jänich, 1984]. Thus, there exists a convergent subsequence $\{(a_j, b_j)\} \subseteq \{(a_i, b_i)\}$. Let $lim_{j\to\infty}(a_j, b_j) = (a, b)$. Since both $\mathcal{A}$ and $\mathcal{B}$ are closed sets, $a \in \mathcal{A}$ and $b \in \mathcal{B}$.

By definition of $\{a_i\}$ and $\{b_i\}$, we have $a_j(\mathbf{q}_j) = b_j \in \mathcal{A}(\mathbf{q}_j) \cap \mathcal{B}$. Hence $\|a_j(\mathbf{q}_j) - b_j\| = 0$, and $\lim_{j\to\infty} \|a_j(\mathbf{q}_j) - b_j\| = 0$. Since the Euclidean distance $\|x - y\|$ is a continuous function of both x and y and $f(a_j, \mathbf{q}_j) = a_j(\mathbf{q}_j)$ is a continuous function of both $\mathbf{q}_j$ and a_j, we can write:

$$\lim_{j\to\infty} \|a_j(\mathbf{q}_j) - b_j\| = \|a(\mathbf{q}) - b\|.$$

Therefore: $\|a(\mathbf{q}) - b\| = 0$, $\mathbf{q} \in \mathcal{CB}$, and $\mathcal{CB}$ is closed.

(2) Let $\mathcal{C}$ be represented as $\mathbf{R}^N \times SO(N)$. Let $\rho = \max_{a\in\mathcal{A}} \|O_{\mathcal{A}} - a\|$ and $ball(\rho, 0)$ be the N-ball of radius ρ centered at the origin of $\mathcal{F}_{\mathcal{W}}$. For all $\Theta \in SO(N)$, we have $\mathcal{CB}_\Theta \subseteq \mathcal{X}$, with $\mathcal{X} = \mathcal{B} \ominus ball(\rho, 0)$. Hence, $\mathcal{CB} \subseteq \mathcal{X} \times SO(N)$. Since $SO(N)$ is compact, $\mathcal{CB}$ is bounded. ∎

PROPOSITION 5: *$\mathcal{CB}$ is closed.*

Proof: This proposition can be established by slightly modifying the first part of the above proof to handle the case where $\mathcal{B}$ is not bounded. Since the sequence $\{\mathbf{q}_i\}$ converges, there exists Q such that the set of all $\mathbf{q}_i$, with $i > Q$, maps in $\mathbf{R}^N \times SO(N)$ to a bounded subset. The area $\bigcup_{i>Q} \mathcal{A}(\mathbf{q}_i) \subset \mathcal{W}$ is bounded, and hence is contained in a compact subset S of $\mathcal{W}$. The sequence $\{b_i\}$ is contained in $\mathcal{B} \cap S$, which is a compact metric set, hence a sequentially compact set. ∎

PROPOSITION 6: *If both $\mathcal{A}$ and $\mathcal{B}$ are connected, so is $\mathcal{CB}$.*

Proof: Let us consider any two configurations $\mathbf{q}_1 = (X_1, \Theta_1)$ and $\mathbf{q}_2 = (X_2, \Theta_2)$ in $\mathcal{CB}$. By definition, $\mathcal{A}(\mathbf{q}_1) \cap \mathcal{B} \neq \emptyset$ and $\mathcal{A}(\mathbf{q}_2) \cap \mathcal{B} \neq \emptyset$. Assume that $\mathcal{A}$ is at configuration $\mathbf{q}_1$. We can construct a path τ such that $\tau(0) = \mathbf{q}_1$, $\tau(1) = \mathbf{q}_2$, and, for all $s \in [0, 1]$, $\tau(s) \in \mathcal{CB}$. This path is the product $\tau = \tau_1 \bullet \tau_2 \bullet \tau_3$ of three paths τ_1, τ_2, and τ_3 defined below.

Let a_1 be a point of $\mathcal{A}$ and b_1 a point of $\mathcal{B}$ such that $a_1(\mathbf{q}_1)$ and b_1 coincide. Let a_2 be a point of $\mathcal{A}$ and b_2 a point of $\mathcal{B}$ such that $a_2(\mathbf{q}_2)$

and b_2 coincide.

- The path τ_1 consists of rotating $\mathcal{A}$ around a fixed axis passing through a_1 from the orientation Θ_1 to the orientation Θ_2.
- Let σ be a continuous curve segment included in $\mathcal{B}$ and connecting b_1 to b_2. τ_2 is obtained when $\mathcal{A}$ moves at the fixed orientation Θ_2 in such a way that a_1 follows σ.
- Let σ' be a continuous curve segment included in $\mathcal{A}$ and connecting a_1 to a_2. τ_3 is obtained when $\mathcal{A}$ moves at the fixed orientation Θ_2 in such a way that b_2 follows σ'.

Thus, $\mathcal{CB}$ is connected. ∎

Notice that the last proof requires that $\mathcal{A}$ can translate freely at any fixed orientation.

9.3 Regularity

In this subsection we assume that both $\mathcal{A}$ and $\mathcal{B}$ are **regular** sets, i.e. $\mathcal{A} = cl(int(\mathcal{A}))$ and $\mathcal{B} = cl(int(\mathcal{B}))$ [Requicha, 1977]. We show that under this assumption, $\mathcal{CB}$ is also regular.

Let $\mathcal{CB}_{overlap}$ denote the subset of configurations where the interior of $\mathcal{A}$ intersects the interior of $\mathcal{B}$, i.e. :

$$\mathcal{CB}_{overlap} = \{\mathbf{q} \in \mathcal{C} \;/\; int(\mathcal{A}(\mathbf{q})) \cap int(\mathcal{B}) \neq \emptyset\}.$$

Let $\mathcal{CB}_{contact}$ denote the subset of configurations where $\mathcal{A}$ and $\mathcal{B}$, but not their interiors, intersect, i.e. :

$$\mathcal{CB}_{contact} = \{\mathbf{q} \in \mathcal{C} \;/\; \mathcal{A}(\mathbf{q}) \cap \mathcal{B} \neq \emptyset \text{ and } int(\mathcal{A}(\mathbf{q})) \cap int(\mathcal{B}) = \emptyset\}.$$

We have: $\mathcal{CB}_{overlap} = \mathcal{CB} \backslash \mathcal{CB}_{contact}$.

The boundary of $\mathcal{CB}$ is part of $\mathcal{CB}_{contact}$. Indeed, since $\mathcal{A}$ and $\mathcal{B}$ are regular, if $\mathbf{q}$ is in $\partial\mathcal{CB}$, then an arbitrarily small displacement of $\mathcal{A}$ can make $\mathcal{A}$ and $\mathcal{B}$ disjoint, which would not be possible if the interiors of $\mathcal{A}(\mathbf{q})$ and $\mathcal{B}$ were intersecting. However, $\partial\mathcal{CB}$ may be different from $\mathcal{CB}_{contact}$, and the only general relation between them is:

$$\partial\mathcal{CB} \subseteq \mathcal{CB}_{contact}.$$

TESLA EXHIBIT 1013, Page 93

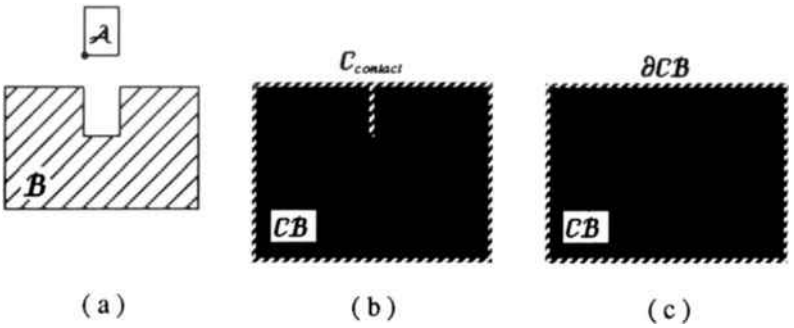


Figure 9. This figure illustrates an example where $\partial\mathcal{CB} \neq \mathcal{C}_{contact}$. Figure a depicts the objects $\mathcal{A}$ and $\mathcal{B}$ in the workspace. $\mathcal{C}_{contact}$ and $\partial\mathcal{CB}$ are shown in striped lines in Figures b and c, respectively.

Figure 9 gives a simple example where $\partial\mathcal{CB}$ is strictly included in $\mathcal{CB}_{contact}$.

PROPOSITION 7: $\mathcal{CB} = cl(\mathcal{CB}_{overlap})$.

Proof: Let $\mathbf{q} = (\mathcal{T}, \Theta)$ be a configuration in $\mathcal{CB}$. Let a be a point in $\mathcal{A}$ and b a point in $\mathcal{B}$ such that $a(\mathbf{q})$ and b coincide in $\mathcal{W}$. Since both $\mathcal{A}$ and $\mathcal{B}$ are regular sets, there are sequences of points $\{a_k\}$ in $int(\mathcal{A})$ and $\{b_k\}$ in $int(\mathcal{B})$ such that $\lim_{k\to\infty} a_k = a$ and $\lim_{k\to\infty} b_k = b$. Let $\{\mathbf{q}_k\} = \{(\mathcal{T}_k, \Theta)\}$ be a sequence of configurations in $\mathcal{CB}_{overlap}$ such that $\mathcal{A}(\mathbf{q}_k)$ and $\mathcal{A}(\mathbf{q})$ have the same orientation Θ and $a_k(\mathbf{q}_k)$ and b_k coincide. We have: $\lim_{k\to\infty} \mathbf{q}_k = \mathbf{q}$. Therefore: $\mathbf{q} \in cl(\mathcal{CB}_{overlap})$ and $\mathcal{CB} \subseteq cl(\mathcal{CB}_{overlap})$.

Since $\mathcal{CB}_{overlap} \subset \mathcal{CB}$ and $\mathcal{CB}$ is closed, we also have: $cl(\mathcal{CB}_{overlap}) \subseteq \mathcal{CB}$.

Hence: $\mathcal{CB} = cl(\mathcal{CB}_{overlap})$. ∎

Notice that this proof requires that $\mathcal{A}$ be able to translate freely at fixed orientation.

PROPOSITION 8: *$\mathcal{CB}$ is regular.*

Proof: The two relations:

$$\mathcal{CB}_{overlap} = \mathcal{CB} \backslash \mathcal{CB}_{contact}$$

and

$$\partial \mathcal{CB} \subseteq \mathcal{CB}_{contact}$$

entail:

$$\mathcal{CB}_{overlap} \subseteq int(\mathcal{CB}) \subset \mathcal{CB}.$$

Taking the closure of each set, we obtain:

$$cl(\mathcal{CB}_{overlap}) \subseteq cl(int(\mathcal{CB})) \subseteq \mathcal{CB}.$$

This relation combined with Proposition 7 implies that $cl(int(\mathcal{CB})) = \mathcal{CB}$. Hence, $\mathcal{CB}$ is regular. ∎

9.4 Free and Semi-Free Paths

A free path is a path along which $\mathcal{A}$ touches or overlaps no obstacle. This is formalized as follows:

DEFINITION 4: Free space *is the subset of* $\mathcal{C}$ *defined by:*

$$\mathcal{C}_{free} = \mathcal{C} \setminus \bigcup_{i \in [1,q]} \mathcal{CB}_i.$$

Any configuration in $\mathcal{C}_{free}$ *is called a* **free configuration**. *A* **free path** *between two free configurations* $\mathbf{q}_{init}$ *and* $\mathbf{q}_{goal}$ *is a continuous map* $\tau : [0,1] \rightarrow \mathcal{C}_{free}$, *with* $\tau(0) = \mathbf{q}_{init}$ *and* $\tau(1) = \mathbf{q}_{goal}$.

Since the C-obstacles are closed subsets of $\mathcal{C}$, $\mathcal{C}_{free}$ is an open subset of $\mathcal{C}$, hence is a smooth manifold.

Two configurations belong to the same connected components of $\mathcal{C}_{free}$ if and only if they are connected by a free path. Thus, the basic motion planning problem — finding a free path between two free configurations — includes the pure topological problem of deciding whether the two configurations belong to the same connected component of $\mathcal{C}_{free}$.

Two free paths τ and τ' are **topologically equivalent** if and only if they are homotopic in $\mathcal{C}_{free}$, i.e. if there exists a continuous map $h : [0,1] \times [0,1] \rightarrow \mathcal{C}_{free}$, with $h(s,0) = \tau(s)$ and $h(s,1) = \tau'(s)$ for all $s \in [0,1]$.

Example: Consider the simple two-dimensional configuration space $\mathcal{C} = \mathbf{R}^2$ shown in Figure 10 with three disjoint compact C-obstacles.

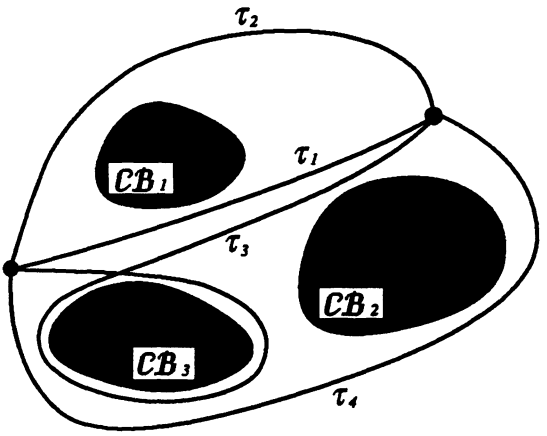


Figure 10. In $C = \mathbf{R}^2$, every compact, connected component of the C-obstacle region induces a countable infinity of classes of topologically equivalent paths. The above figure shows four paths belonging to four distinct classes.

Although $\mathbf{R}^2$ is simply connected, C_{free} is multiply connected in this example. Between any two configurations in C_{free} there is a countable infinity of different classes of topologically equivalent paths. ∎

Any free path is topologically equivalent to a (piecewise) smooth free path that is arbitrarily close to it (see Subsection 6.4).

DEFINITION 5: *A* **semi-free path** *between two configurations* $\mathbf{q}_{init}$ *and* $\mathbf{q}_{goal}$ *is a continuous map* $\tau : [0,1] \rightarrow cl(C_{free})$, *with* $\tau(0) = \mathbf{q}_{init}$ *and* $\tau(1) = \mathbf{q}_{goal}$.

It can be verified that if there exists a semi-free path between two configurations, then there also exists a path between the same two configurations that lies in free space, except perhaps at isolated configurations [Laumond, 1986].

If $cl(C_{free})$ is a *manifold with boundary* [Guillemin and Pollack, 1974] — i.e. if every configuration in $cl(C_{free})$ has a neighborhood homeomorphic to either an open ball or a closed half-space of $\mathbf{R}^m$ — then the existence of a semi-free path τ' between two free configurations entails the existence of a free path τ. One such path can be obtained by an arbitrarily small deformation of τ'.

TESLA EXHIBIT 1013, Page 96

9.5 Contact Space and Valid Space

In this subsection we assume that both $\mathcal{A}$ and the $\mathcal{B}_i$'s are regular sets.

Contact space is the subset of $\mathcal{C}$ made of configurations at which $\mathcal{A}$ touches one or several obstacles without overlapping any. The union of free space and contact space is *valid space*, the set of configurations achievable by $\mathcal{A}$ when we accept contacts between $\mathcal{A}$ and the obstacles.

DEFINITION 6: Contact space *is the subset of $\mathcal{C}$ defined as:*

$$\mathcal{C}_{contact} = \{\mathbf{q} \in \mathcal{C} \,/\, \mathcal{A}(\mathbf{q}) \cap \bigcup_{i=1}^{q} \mathcal{B}_i \neq \emptyset \text{ and } int(\mathcal{A}(\mathbf{q})) \cap \bigcup_{i=1}^{q} int(\mathcal{B}_i) = \emptyset\}.$$

Valid space *is:* $\mathcal{C}_{valid} = \mathcal{C}_{free} \cup \mathcal{C}_{contact}$.

One can easily verify that $\mathcal{C}_{valid}$ is a closed subset of $\mathcal{C}$ and that $\partial(\bigcup_i \mathcal{CB}_i) \subseteq \mathcal{C}_{contact}$. Related properties include: $\partial\mathcal{C}_{free} \subseteq \mathcal{C}_{contact}$, $cl(\mathcal{C}_{free}) \subseteq \mathcal{C}_{valid}$, and $\partial\mathcal{C}_{free} \subseteq \partial\mathcal{C}_{valid}$.

DEFINITION 7: *A* **valid path** *(resp.* **contact path***) between two configurations $\mathbf{q}_{init}$ and $\mathbf{q}_{goal}$ in $\mathcal{C}_{valid}$ (resp. $\mathcal{C}_{contact}$) is a continuous map $\tau : [0,1] \rightarrow \mathcal{C}_{valid}$ (resp. $\mathcal{C}_{contact}$), with $\tau(0) = \mathbf{q}_{init}$ and $\tau(1) = \mathbf{q}_{goal}$.*

Let $\mathcal{A}$ and $\mathcal{B}$ be two connected objects. It can be shown, under the condition that $\mathcal{A}$ is free-flying, that if there exists a valid path of $\mathcal{A}$ between two contact configurations where $\mathcal{A}$ touches $\mathcal{B}$, then there also exists a contact path between the same two configurations [Hopcroft and Wilfong, 1986a].

10 Representation of Forces

Forces play no role in the definition of the basic motion planning problem. However, an important planning approach, potential field, uses artificial forces to guide planing. Forces will also be considered in Chapter 10 when we discuss motion planning with uncertainty in control.

Forces exerted on $\mathcal{A}$ at configuration $\mathbf{q}$ can be represented as vectors in the tangent space $T_{\mathbf{q}}(\mathcal{C})$. This representation requires that an inner

product be defined in the tangent spaces, and it depends on the choice of this inner product. This results from the fact that a force is defined through the *virtual work* that it produces along virtual displacements [Greenwood, 1965]. This work, which is independent of the representation of the force, is expressed as an inner product. Changing the inner product requires us to change the representation of the force.

Let $\vec{\mathbf{F}}(\mathbf{x})$ be a field of N-dimensional vectors representing a field of forces in $\mathcal{W}$. The work of these forces along a curve $\gamma : s \in [0,1] \mapsto \gamma(s) \in \mathcal{W}$ is:

$$\int_0^1 \vec{\mathbf{F}}(\gamma(s)) \cdot \frac{d\vec{\gamma}(s)}{ds} ds$$

where "$\cdot$" denotes the inner product in $\mathcal{W}$ and $\frac{d\vec{\gamma}(s)}{ds}$ is the tangent vector to γ at s.

Assume that the forces $\vec{\mathbf{F}}(\mathbf{x})$ are applied only at point a of $\mathcal{A}$. Thus, when $\mathcal{A}$ is at configuration $\mathbf{q}$, the force $\vec{\mathbf{F}}(a(\mathbf{q}))$ is applied to $\mathcal{A}$ at a. Let "$<\ ,\ >$" denote the inner product placed in $T_{\mathbf{q}}(\mathcal{C})$. The mapping of $\vec{\mathbf{F}}(\mathbf{x})$ in $T_{\mathbf{q}}(\mathcal{C})$ is defined as the vector field $\vec{F}(\mathbf{q}) \in T_{\mathbf{q}}(\mathcal{C})$ such that:

$$< \vec{F}(\mathbf{q}), \vec{\delta t} > = \vec{\mathbf{F}}(a(\mathbf{q})) \cdot \vec{\delta \mathbf{x}}$$

where $\vec{\delta t}$ denotes any virtual displacement of $\mathcal{A}$ along any vector $\vec{t} \in T_{\mathbf{q}}(\mathcal{C})$ and $\vec{\delta \mathbf{x}}$ denotes the corresponding virtual displacement of a in $\mathcal{W}$. This relation expresses that the work of $\vec{\mathbf{F}}$ along $\vec{\delta \mathbf{x}}$ is equal to the work of $\vec{F}$ along $\vec{\delta t}$. Hence, for any path $\tau : s \in [0,1] \mapsto \tau(s) \in \mathcal{C}$, the work of $\vec{F}$ along τ is equal to the work of $\vec{\mathbf{F}}$ along the path $\tau_a : s \in [0,1] \mapsto a(\tau(s)) \in \mathcal{W}$ followed by a when $\mathcal{A}$ moves along τ, i.e. :

$$\int_0^1 < \vec{F}(\tau(s)), \frac{d\vec{\tau}(s)}{ds} > ds = \int_0^1 \vec{\mathbf{F}}(\tau_a(s)) \cdot \frac{d\vec{\tau}_a(s)}{ds} ds.$$

Below, we consider that $\mathcal{A}$ is at configuration $\mathbf{q}$; we abbreviate $\vec{F}(\mathbf{q})$ to $\vec{F}$ and $\vec{\mathbf{F}}(a(\mathbf{q}))$ to $\vec{\mathbf{F}}$.

Consider the case where $N = 3$. Assume that the inner product $<\ ,\ >$ in $T_{\mathbf{q}}(\mathcal{C})$ makes the principal axis basis into an orthonormal basis, i.e. if $(\xi_1 ... \xi_m)^T$ and $(\zeta_1 ... \zeta_m)^T$ are vectors $\vec{\xi}$ and $\vec{\zeta}$ of $T_{\mathbf{q}}(\mathcal{C})$ described by their

components in the principal axis basis, the inner product of $\vec{\xi}$ and $\vec{\zeta}$ is:

$$< \vec{\xi}, \vec{\zeta} > = \begin{pmatrix} \xi_1 & \cdots & \xi_m \end{pmatrix} \begin{pmatrix} \zeta_1 \\ \cdot \\ \cdot \\ \cdot \\ \zeta_m \end{pmatrix}.$$

Let $\vec{F}$ be the vector of $T_{\mathbf{q}}(\mathcal{C})$ whose components in the principal axis basis are $f_1, f_2, f_3, m_1, m_2, m_3$, with f_i, $i = 1,2,3$, being the components of $\vec{\mathbf{F}}$ along the x-, y-, and z-axes of $\mathcal{F}_{\mathcal{A}}$, and m_j, $j = 1,2,3$, being the components of the vector $\vec{O_{\mathcal{A}}a} \wedge \vec{\mathbf{F}}$ along these axes. Let $(\delta x_{\mathcal{A}}\ \delta y_{\mathcal{A}}\ \delta z_{\mathcal{A}}\ \delta\alpha\ \delta\beta\ \delta\gamma)^T$ be any virtual displacement $\vec{\delta t}$ of $\mathcal{A}$ expressed in the principal axis basis. The corresponding displacement $\vec{\delta \mathbf{x}}$ of a is the 3-vector:

$$\begin{pmatrix} \delta x_{\mathcal{A}} \\ \delta y_{\mathcal{A}} \\ \delta z_{\mathcal{A}} \end{pmatrix} + \begin{pmatrix} \delta\alpha \\ \delta\beta \\ \delta\gamma \end{pmatrix} \wedge \vec{O_{\mathcal{A}}a}.$$

We have:

$$< \vec{F}, \vec{\delta t} > = \vec{\mathbf{F}} \cdot \vec{\delta \mathbf{x}}.$$

Therefore, the six-dimensional vector $\vec{F}$ defined as above represents the force $\vec{\mathbf{F}}$ in the principal axis basis of $T_{\mathbf{q}}(\mathcal{C})$.

In a similar fashion, if $N = 2$, the force $\vec{\mathbf{F}}$ maps in $T_{\mathbf{q}}(\mathcal{C})$ to a 3-vector $\vec{F}$, whose components in the principal axis basis of the tangent space are f_1, f_2 and m, with f_1 and f_2 being the components of $\vec{\mathbf{F}}$ along the x- and y-axes of $\mathcal{F}_{\mathcal{A}}$ and $m = \vec{O_{\mathcal{A}}a} \wedge \vec{\mathbf{F}}$.

Consider now two distinct inner products in $T_{\mathbf{q}}(\mathcal{C})$, denoted by $< , >_1$ and $< , >_2$. Let β_1 and β_2 be two bases of $T_{\mathbf{q}}(\mathcal{C})$ such that β_i, $i = 1,2$, is an orthonormal basis when the inner product $< , >_i$ is used. Let $\vec{t}$ be any vector in $T_{\mathbf{q}}(\mathcal{C})$. We denote by $[\vec{t}\,]_i$ the m-vector of the components of $\vec{t}$ in β_i. J denotes the $m \times m$ matrix such that:

$$[\vec{t}\,]_2 = J\,[\vec{t}\,]_1.$$

(In the case where the bases β_1 and β_2 are induced by charts in $\mathcal{C}$, say (U_1, ϕ_1) and (U_2, ϕ_2), J is the Jacobian matrix of the map $\phi_2 \circ \phi_1^{-1}$.)

Let the same force be represented by two vectors: $\vec{F}_1$ when $< \ , \ >_1$ is used, and $\vec{F}_2$ when $< \ , \ >_2$ is used. For all virtual vectors $\vec{\delta t} \in T_{\mathbf{q}}(\mathcal{C})$, we have:

$$< \vec{F}_1, \vec{\delta t} >_1 = < \vec{F}_2, \vec{\delta t} >_2$$

since both expressions represent the same virtual work. Hence:

$$[\ \vec{F}_1\ ]_1^T\ [\ \vec{\delta t}\ ]_1 = [\ \vec{F}_2\ ]_2^T\ [\ \vec{\delta t}\ ]_2.$$

Using the relation $[\ \vec{\delta t}\ ]_2 = J\ [\ \vec{\delta t}\ ]_1$, we obtain:

$$[\ \vec{F}_1\ ]_1^T\ [\ \vec{\delta t}\ ]_1 = [\ \vec{F}_2\ ]_2^T\ J\ [\ \vec{\delta t}\ ]_1$$

which leads to:

$$[\ \vec{F}_1\ ]_1 = J^T\ [\ \vec{F}_2\ ]_2.$$

Let (U, ϕ) be a chart in $\mathcal{C}$. Let $g : \mathbf{R}^m \rightarrow \mathbf{R}^N$ be the application that maps the coordinates x_i, $i = 1, ..., m$, of $\mathbf{q}$ in the chart to the N coordinates of $a(\mathbf{q})$ in $\mathcal{W}$. Assume that the inner product in $T_{\mathbf{q}}(\mathcal{C})$ makes the basis β induced by the chart (U, ϕ) into an orthonormal basis. Let J_g be the Jacobian matrix of g, i.e. the $N \times m$ matrix $\left(\frac{dg_i}{dx_j}\right)$, $i = 1, ..., N$, $j = 1, ..., m$, of first derivatives. From:

$$< \vec{F}, \vec{\delta t} > = \ \vec{\mathbf{F}} \cdot \vec{\delta \mathbf{x}}$$

we obtain:

$$[\ \vec{F}\ ]_\beta^T \ = \ (f_1 \ ... \ f_N)\ J_g$$

and:

$$[\ \vec{F}\ ]_\beta \ = \ J_g^T \begin{pmatrix} f_1 \\ . \\ . \\ . \\ f_N \end{pmatrix}$$

where f_1,...,f_N denote the components of $\vec{\mathbf{F}}$ in $\mathcal{F}_\mathcal{W}$.

An interesting case occurs when the force $\vec{\mathbf{F}}$ is the reaction force exerted by an obstacle $\mathcal{B}$ on $\mathcal{A}$. For simplification, we assume that $a \in \mathcal{A}$ is the only contact point of $\mathcal{A}$ with the obstacles and that the contact is between a vertex X of $\mathcal{A}$ (or $\mathcal{B}$) and a smooth $(N-1)$-dimensional subset S of $\mathcal{B}$'s (or $\mathcal{A}$'s) boundary. We denote the configuration of $\mathcal{A}$

TESLA EXHIBIT 1013, Page 100

by $\mathbf{q}_c \in \mathcal{C}_{contact}$. Let (U, ϕ) be any chart of $\mathcal{C}$ such that $\mathbf{q}_c \in U$. U is taken sufficiently small so that ϕ maps $\mathcal{C}_{contact} \cap U$ to a region of $\mathbf{R}^m$ determined by a single equation $f(x_1, ..., x_m) = 0$, where f is a smooth function of the variables $x_1, ..., x_m$, expressing that X is in S. Thus, $\mathcal{C}_{contact} \cap U$ is diffeomorphic to an open subset of $\mathbf{R}^{m-1}$. Let $\mathcal{A}$ move slightly in such a way that X stays in contact with S. This motion can be described as a differentiable curve γ_c in $\mathcal{C}_{contact} \cap U$. The tangent vectors of all the possible curves γ_c at $\mathbf{q}_c$ span an $(m-1)$-subspace of $T_{\mathbf{q}_c}(\mathcal{C})$, which is nothing other than the tangent space $T_{\mathbf{q}_c}(\mathcal{C}_{contact})$ of $\mathcal{C}_{contact}$ at $\mathbf{q}_c$. Assuming that the contact between X and S is frictionless, the work of the reaction force when $\mathcal{A}$ moves along γ_c is zero. Therefore, whatever inner product is used in $T_{\mathbf{q}}(\mathcal{C})$, the reaction force exerted by $\mathcal{B}$ on $\mathcal{A}$ maps in $T_{\mathbf{q}_c}(\mathcal{C})$ to a vector perpendicular to $T_{\mathbf{q}_c}(\mathcal{C}_{contact})$.

Exercises

1: Let $\mathcal{F}_{\mathcal{W}}$ and $\mathcal{F}_{\mathcal{A}}$ be two Cartesian frames in $\mathbf{R}^N$, $N = 2$ or 3. Assume that the origins of the two frames coincide, but not their axes. Let Θ be the $N \times N$ matrix whose columns are the components, in $\mathcal{F}_{\mathcal{W}}$, of the unit vectors along the axes of $\mathcal{F}_{\mathcal{A}}$. Show that Θ can be interpreted as a rotational operator that moves the axes of $\mathcal{F}_{\mathcal{W}}$ onto those of $\mathcal{F}_{\mathcal{A}}$. Let Θ_1 and Θ_2 be two matrices specified as Θ. Which rotational operators do the matrix products $\Theta_1 \times \Theta_2$ and $\Theta_2 \times \Theta_1$ represent?

2: Let $\mathcal{F}_{\mathcal{W}}$ and $\mathcal{F}_{\mathcal{A}}$ be two Cartesian frames in $\mathbf{R}^N$, $N = 2$ or 3. Let $O_{\mathcal{W}}$ and $O_{\mathcal{A}}$ denote the origins of $\mathcal{F}_{\mathcal{W}}$ and $\mathcal{F}_{\mathcal{A}}$, respectively. Let $\mathcal{T}$ be the N-vector of the coordinates of $O_{\mathcal{A}}$ in $\mathcal{F}_{\mathcal{W}}$. Let Θ be the $N \times N$ matrix whose columns are the components, in $\mathcal{F}_{\mathcal{W}}$, of the unit vectors along the axes of $\mathcal{F}_{\mathcal{A}}$. Using $\mathcal{T}$ and Θ, express the relation between the coordinates of any point of $\mathbf{R}^N$ in $\mathcal{F}_{\mathcal{W}}$ and its coordinates in $\mathcal{F}_{\mathcal{A}}$.

3: Prove that any matrix in $SO(3)$ determines a single rotation through an angle θ about some axis $\vec{n}$.

4: Show that $SO(2)$ is a commutative (abelian) group. Show that $SO(3)$ is a non-commutative group.

5: The configuration space of a rigid object in $\mathcal{W} = \mathbf{R}^N$, $N = 2$ or 3,

is $\mathcal{C} = \mathbf{R}^N \times SO(N)$. Its dimension is $m = \frac{1}{2}N(N+1)$. Explain why representing $\mathcal{C}$ as a single copy of $\mathbf{R}^m$ (or a subset of $\mathbf{R}^m$) does not allow us to model $\mathcal{A}$'s motions adequately?

6: Consider the distance function $d : \mathcal{C} \times \mathcal{C} \rightarrow \mathbf{R}$ defined as:

$$d(\mathbf{q}_1, \mathbf{q}_2) = \max_{a \in \mathcal{A}} \|a(\mathbf{q}_1) - a(\mathbf{q}_2)\|.$$

Verify that it induces the same topology of $\mathcal{C}$ as the Euclidean metric in the ambient space $\mathbf{R}^M$.

7: Consider the subset S of $\mathbf{R}^2$ given by:

$$S = \{(x, \sin\frac{1}{x}) \,/\, 0 < x \leq 1\} \cup \{(0, y) \,/\, y \in [-1, +1]\}.$$

Are any two points in S connected by a path in S?

8: Show that infinitesimal rotations form a vector space. [Hint: Consider two matrices Θ_1 and Θ_2 of $SO(3)$ representing rotations about two vectors $\vec{n}_1$ and $\vec{n}_2$ through infinitesimal angles $d\theta_1$ and $d\theta_2$, respectively. Compute $\Theta_1 \times \Theta_2$ and $\Theta_2 \times \Theta_1$.] Relate this result to the fact that $\mathcal{C}$ is a manifold.

9: Verify that the following relation (see Subsection 4.3) holds:

$$\forall a \in \mathcal{A} : a(\mathbf{q}) = \mathbf{X}_\Theta a(0) \mathbf{X}_\Theta^* + \mathcal{T}$$

10: Consider S^3, the unit 3-sphere in $\mathbf{R}^4$ that represents the space of unit quaternions. Let $\mathbf{X}$ be a unit quaternion. Show that the curve in S^3 representing a rotation of 4π about a fixed axis begins and ends at the same point $\mathbf{X}$ of S^3, and that the curve representing a rotation of 2π begins and ends at two antipodal points $\mathbf{X}$ and $-\mathbf{X}$.

11: Define the stereographic projection mapping the sphere S^3 onto $\mathbf{R}^3$ as a way to algebraically represent elements of $SO(3)$.

12: Verify that homotopy determines a relation of equivalence on the set of all paths having the same initial configuration and the same final configuration.

TESLA EXHIBIT 1013, Page 102

13: Let $\tau_1 \bullet \tau_2$ denote the product of two paths τ_1 and τ_2, and $\overline{\tau}$ the inverse of the path τ (see Subsection 5.1).

a. Show that if τ_1 and τ_1' are homotopic and if τ_2 and τ_2' are homotopic, then $\tau_1 \bullet \tau_2$ and $\tau_1' \bullet \tau_2'$ are also homotopic.

b. Show that if two paths τ_1 and τ_2 are homotopic, the loop $\tau_1 \bullet \overline{\tau_2}$ is null-homotopic.

c. Show that if the loop $\tau_1 \bullet \overline{\tau_2}$ is null-homotopic, then the two paths τ_1 and τ_2 are homotopic. [Hint: Consider the two paths τ_1 and $\tau_1 \bullet \overline{\tau_2} \bullet \tau_2$.]

14: Let $s \in \mathbf{R} \mapsto \gamma(s) \in \mathcal{C}$ be a curve in the smooth manifold $\mathcal{C}$. Show that the property of γ of being differentiable at $s = s_0$ is independent of the selected chart in which the property is verified.

15: Show that if a path τ is piecewise of class C^1, it has a finite length in any metric inducing the topology of $\mathcal{C}$.

16: Consider a two-dimensional polygonal object $\mathcal{A}$ translating in the plane at fixed orientation among polygonal obstacles. With an example, show that different polygonal shapes for both $\mathcal{A}$ and the obstacles can yield C-obstacles with the same geometry in $\mathcal{C} = \mathbf{R}^2$.

17: Show that the Minkowski sum $E \oplus F$ of two convex subsets of $\mathbf{R}^N$ is a convex subset of $\mathbf{R}^N$.

18: Let $\mathcal{A}$ be a free-flying object in $\mathcal{W} = \mathbf{R}^2$. Its configuration space is $\mathcal{C} = \mathbf{R}^2 \times S^1$. Configurations are parameterized by $\mathbf{q} = (x, y, \theta)$, with $x \in \mathbf{R}$, $y \in \mathbf{R}$, and $\theta \in [0, 2\pi)$. x and y denote the coordinates of $O_{\mathcal{A}}$ in $\mathcal{F}_{\mathcal{W}}$. θ denotes the angle (mod 2π) between the x-axes of $\mathcal{F}_{\mathcal{W}}$ and $\mathcal{F}_{\mathcal{A}}$.

Let $\mathcal{B}$ be an obstacle in $\mathcal{W}$. $\mathcal{CB}$ is the corresponding C-obstacle in $\mathcal{C}$.

Let $\mathcal{A}'$ be the region of $\mathbf{R}^2$ swept out by $\mathcal{A}$ when it rotates around $O_{\mathcal{A}}$ from the orientation θ_1 to the orientation θ_2. Let $O_{\mathcal{A}'}$ be the point of $\mathcal{A}'$ that coincides with $O_{\mathcal{A}}$ during the rotation. We treat $\mathcal{A}'$ as an object that can only translate in $\mathbf{R}^2$. We represent a configuration of $\mathcal{A}'$ as the coordinates of $O_{\mathcal{A}'}$ in $\mathcal{F}_{\mathcal{W}}$. The configuration space of $\mathcal{A}'$ is $\mathcal{C}' = \mathbf{R}^2$ and the C-obstacle corresponding to $\mathcal{B}$ in this space is denoted by $\mathcal{CB}'$.

TESLA EXHIBIT 1013, Page 103

$\pi_{[\theta_1,\theta_2]}(\mathcal{CB})$ denotes the projection of the slice of the volume $\mathcal{CB}$ comprised between $\theta = \theta_1$ and $\theta = \theta_2$ on the xy-plane.

Prove that: $\pi_{[\theta_1,\theta_2]}(\mathcal{CB}) = \mathcal{CB}'$.

19: Show that every path in the configuration space of a free-flying rigid object is homotopic to a path that is the product of a finite number of paths, each being either a pure translation or a pure rotation of the object. Show that if two configurations are connected by a free path, they are also connected by a free path made of a finite succession of pure translations and pure rotations.

20: Define the subset $\mathcal{C}^{\varepsilon}_{free}$ of $\mathcal{C}$ made of all configurations at which the robot $\mathcal{A}$ is distant from all the obstacles $\mathcal{B}_i$ by more than ε, with $\varepsilon > 0$. A path in $\mathcal{C}^{\varepsilon}_{free}$ is said to be *ε-free*. Discuss the usefulness of this notion in practical path planning problems.

21: Consider the closure $cl(\mathcal{C}_{free})$ of $\mathcal{C}_{free}$. Let $\mathbf{q}_1$ and $\mathbf{q}_2$ be two configurations such that there exists a semi-free path $\tau' : [0,1] \rightarrow cl(\mathcal{C}_{free})$, with $\tau'(0) = \mathbf{q}_1$ and $\tau'(1) = \mathbf{q}_2$. Show that there exists $\tau : [0,1] \rightarrow cl(\mathcal{C}_{free})$, with $\tau(0) = \mathbf{q}_1$ and $\tau(1) = \mathbf{q}_2$, such that $\forall s \in [0,1] : \tau(s) \in \mathcal{C}_{free}$, except maybe at isolated values of s [Laumond, 1986]. Illustrate this result with an example in $\mathcal{C} = \mathbf{R}^2$.

22: Show that if $cl(\mathcal{C}_{free})$ is a manifold with boundary, then the existence of a semi-free path between any two free configurations entails the existence of a free path between these configurations.

23: Let $\mathcal{A}$ and $\mathcal{B}$ be two convex polygons. Assume that $\mathcal{A}$ can translate freely at fixed orientation. How the reaction force exerted by $\mathcal{B}$ on $\mathcal{A}$, when they are in contact, maps in configuration space (assume no friction)? Justify your answer.

TESLA EXHIBIT 1013, Page 104

Chapter 3

Obstacles in Configuration Space

Obstacles in the workspace $\mathcal{W}$ map in the configuration space $\mathcal{C}$ to regions called C-obstacles. In Chapter 2 we defined the C-obstacle $\mathcal{CB}$ corresponding to a workspace obstacle $\mathcal{B}$ as the following region in $\mathcal{C}$:

$$\mathcal{CB} = \{\mathbf{q} \in \mathcal{C} \,/\, \mathcal{A}(\mathbf{q}) \cap \mathcal{B} \neq \emptyset\}.$$

We also established general set-theoretic and topological properties of C-obstacles. In this chapter we construct the representation of the C-obstacles given that of the workspace objects, i.e. the robot $\mathcal{A}$ and the obstacles.

In Section 1 we analyze in detail the case where the workspace is two--dimensional and the objects are modeled as polygonal regions. We show that when the robot can only translate, the C-obstacles are also polygonal regions of $\mathcal{C}$. Each C-obstacle edge corresponds to the contact between an edge of $\mathcal{A}$ and an obstacle vertex, or the contact between a vertex of $\mathcal{A}$ and an obstacle edge. When $\mathcal{A}$ is allowed to rotate, the C-obstacles become three-dimensional regions bounded by patches of curved ruled surfaces generated by translating and rotating lines.

In Section 2 we treat the case where the workspace is three-dimensional and the objects are modeled as polyhedral regions. The main difference with the polygonal case comes from the existence of three types of contact (face-vertex, vertex-face, and edge-edge), instead of two. When $\mathcal{A}$ is allowed to rotate freely, the C-obstacles are six-dimensional regions bounded by patches of curved surfaces generated by translating and rotating planes.

Finally, in Section 3 we present the more general case where the objects in the workspace are modeled as arbitrary semi-algebraic regions, i.e. are defined by boolean combinations of polynomial equations and inequations, with rational coefficients.

Thus, throughout the chapter, we only consider algebraic models of objects. As a matter of fact, these models are particularly attractive. They give flexibility for describing any physical objects at any desired precision with a reasonably small number of parameters, and they admit a very simple and useful particular case — the polygonal/polyhedral representation. Algebraic models also make it possible to use standard mathematical tools, exact calculation techniques (see [Akritas, 1980] [Schwartz and Sharir, 1983b]) and algebraic decision methods (e.g. the Collins decomposition [Collins, 1975]), which are useful for building motion planning methods. Furthermore, in several circumstances, algebraic models are directly available from Computer-Aided Design systems.

Not surprisingly, most of the current results in motion planning use semi-algebraic representations. It is possible, however, that other models, for example bitmap representation, will attract more interest in the future than they currently do. In Chapter 7 (Potential Field) we will describe an efficient path planner that works from a bitmap representation of the workspace objects.

1 Case of a Polygonal Workspace

In this section we examine the case where $\mathcal{A}$ is a polygon moving freely in the plane among obstacles modeled as polygonal regions. We construct two different representations of the C-obstacles. One representation is in the form of a predicate `CB` taking a configuration $\mathbf{q}$ as argument. `CB`($\mathbf{q}$) evaluates to `true` if and only if $\mathbf{q} \in \mathcal{CB}$, where $\mathcal{CB}$ is the C-obstacle region

represented by CB. The other representation is an explicit description of the boundary of $\mathcal{CB}$ in terms of (curved) faces, edges and vertices, and their adjacency relation. For both representations, we also analyze the particular case where $\mathcal{A}$ can only translate.

1.1 Polygonal Regions and Polygons

An infinite straight line L in $\mathbf{R}^2$ decomposes the plane into two half-planes. Let $h(x,y) = 0$ be the equation of L. We denote by $\overline{h}^-$ and $\overline{h}^+$ the two closed half-planes determined by $h(x,y) \leq 0$ and $h(x,y) \geq 0$, respectively.

DEFINITION 1: *A* **convex polygonal region** *in* $\mathbf{R}^2$ *is the intersection of a finite number of closed half-planes. A* **polygonal region** *is any subset of* $\mathbf{R}^2$ *obtained by taking the union of a finite number of convex polygonal regions.*

A (convex) polygonal region may not be bounded, nor connected. Its connected components may not be simply-connected.

DEFINITION 2: *A* **polygon** *is any polygonal region that is homeomorphic to the closed unit disc.*

Hence, a polygon is a compact, regular, simply-connected region of $\mathbf{R}^2$. Its boundary is a closed-loop sequence of line segments forming a Jordan curve, i.e. a curve homeomorphic to the unit circle.

Let $\mathcal{P}$ be a regular polygonal region. Each maximal connected line segment in $\mathcal{P}$'s boundary that is not intersected by the rest of $\mathcal{P}$'s boundary, except at its endpoints, is called an **edge**. Every endpoint of an edge is called a **vertex**. An edge is said to be **closed** (resp. **open**) if we include (resp. do not include) its endpoints. If $\mathcal{P}$ is a manifold with boundary, each vertex is the endpoint of exactly two edges.

Every edge E_i of $\mathcal{P}$ is contained in an infinite line L_i called the **supporting line** of E_i. Let $h_i(x,y) = 0$ be the equation of L_i. Without loss of generality, we write this equation so that every point in the open edge E_i has a neighborhood whose intersection with $\mathcal{P}$ only contains points (x,y) such that $h_i(x,y) \leq 0$. The half-plane $\overline{h_i}^+$ (resp. $\overline{h_i}^-$) is called the **outer**

(resp. **inner**) half-plane of E_i. The **outgoing** (resp. **ingoing**) **normal** of E_i is the unit vector normal to E_i pointing toward the interior of the outer (resp. inner) half-plane of E_i.

A polygon is conveniently represented by the list of its vertices in counterclockwise order. Each vertex is described by its coordinates. A convex polygonal region may also be represented by the set of inequalities defining the half-planes $\overline{h_i^-}$, i.e. $\bigcap_i \overline{h_i^-}$.

The **complexity** of a polygonal region is defined as the number of its edges and vertices.

1.2 Type A and Type B Contacts

In this subsection we consider an obstacle $\mathcal{B}$ and we construct the equations of the surfaces bounding the corresponding C-obstacle $\mathcal{CB}$. We assume that both $\mathcal{A}$ and $\mathcal{B}$ are convex polygons.

The vertices of $\mathcal{A}$ are denoted by a_i, $i = 1, ..., n_\mathcal{A}$, and those of $\mathcal{B}$ by b_j, $j = 1, ..., n_\mathcal{B}$. The edge of $\mathcal{A}$ connecting a_i to a_{i+1} and the edge of $\mathcal{B}$ connecting b_j to b_{j+1} are denoted by $E_i^\mathcal{A}$ and $E_j^\mathcal{B}$, respectively. (We apply modulo $n_\mathcal{A}$ arithmetic to index i and modulo $n_\mathcal{B}$ arithmetic to index j, so that $a_{n_\mathcal{A}+1} \equiv a_1$ and $b_{n_\mathcal{B}+1} \equiv b_1$.) All edges are treated as closed edges, i.e. they include their endpoints.

We denote the outgoing normal of $E_i^\mathcal{A}(\mathbf{q})$ (resp. $E_j^\mathcal{B}$) by $\vec{\nu}_i^\mathcal{A}(\mathbf{q})$ (resp. $\vec{\nu}_j^\mathcal{B}$). If $\mathbf{q}$ is expressed as $(\mathcal{T}, \Theta)$, where Θ specifies the orientation of $\mathcal{F}_\mathcal{A}$ with respect to $\mathcal{F}_\mathcal{W}$, the outgoing normal of $E_i^\mathcal{A}(\mathbf{q})$ depends only on the orientation Θ of $\mathcal{A}$ at $\mathbf{q}$, though we denote it by $\vec{\nu}_i^\mathcal{A}(\mathbf{q})$.

Let us now suppose that $\mathcal{A}$ and $\mathcal{B}$ are in contact, i.e. :

$$\mathcal{A}(\mathbf{q}) \cap \mathcal{B} \neq \emptyset \quad \text{and} \quad int(\mathcal{A}(\mathbf{q})) \cap int(\mathcal{B}) = \emptyset.$$

We distinguish two types of contact [Lozano-Pérez, 1983] [Donald, 1984]. When an edge $E_i^\mathcal{A}$ of $\mathcal{A}$ contains a vertex b_j of $\mathcal{B}$, the contact is called a **type A contact** (Figure 1). When a vertex a_i of $\mathcal{A}$ is contained in an edge $E_j^\mathcal{B}$ of $\mathcal{B}$, the contact is called a **type B contact** (Figure 2). When the contact zone $\mathcal{A}(\mathbf{q}) \cap \mathcal{B}$ contains both a vertex of $\mathcal{A}$ and a vertex of $\mathcal{B}$, the type of the contact is both A and B.

TESLA EXHIBIT 1013, Page 108

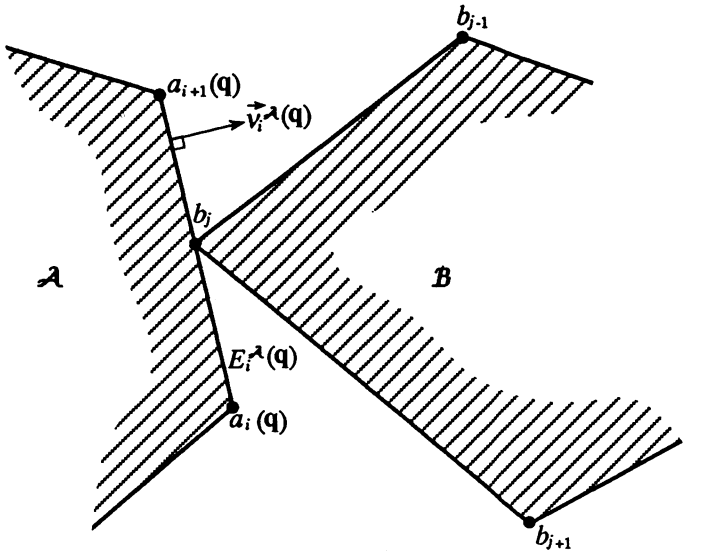


Figure 1. A type A contact between $\mathcal{A}$ and an obstacle $\mathcal{B}$ occurs when an edge $E_i^{\mathcal{A}}$ of $\mathcal{A}$ contains a vertex b_j of $\mathcal{B}$, without overlapping of the interiors of $\mathcal{A}$ and $\mathcal{B}$.

Type A Contact. The constraint that the interiors of $\mathcal{A}$ and $\mathcal{B}$ do not overlap makes a type A contact between $E_i^{\mathcal{A}}$ and b_j feasible only for a subrange of orientations of $\mathcal{A}$. This subrange is determined by the following two conditions (see Figure 1):

$$\vec{\nu}_i^{\mathcal{A}}(\mathbf{q}) \cdot (b_{j-1} - b_j) \geq 0$$

$$\vec{\nu}_i^{\mathcal{A}}(\mathbf{q}) \cdot (b_{j+1} - b_j) \geq 0$$

where $b_{j-1} - b_j$ is the vector connecting b_j to b_{j-1}, $b_{j+1} - b_j$ is the vector connecting b_j to b_{j+1}, and $\vec{u} \cdot \vec{v}$ is the inner product of two vectors in $\mathcal{W}$. We denote the conjunction of these two conditions by $\texttt{APPL}_{i,j}^{\mathcal{A}}(\mathbf{q})$, i.e. :

$$\texttt{APPL}_{i,j}^{\mathcal{A}}(\mathbf{q}) = [\vec{\nu}_i^{\mathcal{A}}(\mathbf{q}) \cdot (b_{j-1} - b_j) \geq 0] \bigwedge [\vec{\nu}_i^{\mathcal{A}}(\mathbf{q}) \cdot (b_{j+1} - b_j) \geq 0].$$

It is called the **applicability condition** of the type A contact between $E_i^{\mathcal{A}}$ and b_j. When $\mathbf{q}$ is expressed as $(\mathcal{T}, \Theta)$, this condition depends only on Θ.

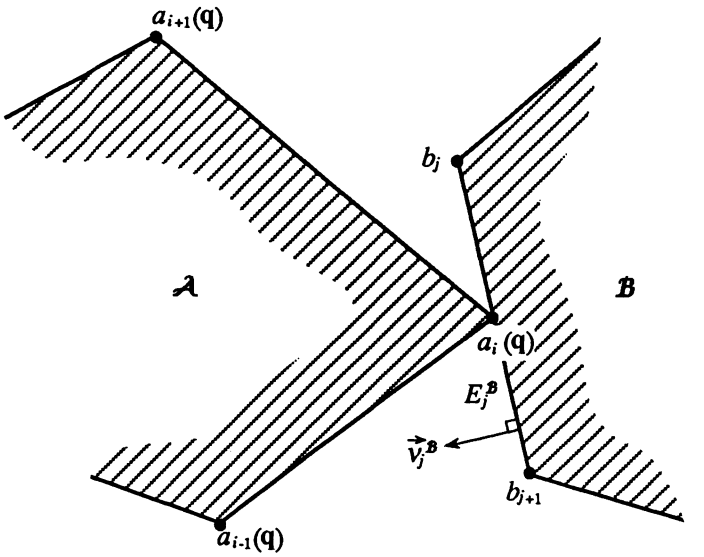


Figure 2. A type B contact between $\mathcal{A}$ and an obstacle $\mathcal{B}$ occurs when a vertex a_i of $\mathcal{A}$ is contained in an edge $E_j^{\mathcal{B}}$ of $\mathcal{B}$, without overlapping of the interiors of $\mathcal{A}$ and $\mathcal{B}$.

Let us displace $\mathcal{A}$ in such a way that the type A contact between $E_i^{\mathcal{A}}$ and b_j is maintained. During the displacement, the robot's configuration moves along a surface in $\mathcal{C}$ (i.e. a two-dimensional submanifold of $\mathcal{C}$) whose equation is:

$$f_{i,j}^{A}(\mathbf{q}) = 0$$

with:

$$f_{i,j}^{A}(\mathbf{q}) = \vec{\nu}_i^{\mathcal{A}}(\mathbf{q}) \cdot (b_j - a_i(\mathbf{q})).$$

This equation expresses the fact that when b_j is in $E_i^{\mathcal{A}}(\mathbf{q})$, the two vectors $\vec{\nu}_i^{\mathcal{A}}(\mathbf{q})$ and $(b_j - a_i(\mathbf{q}))$ are perpendicular or $b_j - a_i(\mathbf{q}) = 0$. The surface it represents is called a **C-surface** of type A. It separates $\mathcal{C}$ into two half-spaces. The C-obstacle $\mathcal{CB}$ lies completely within the closed half-space determined by $f_{i,j}^{A}(\mathbf{q}) \leq 0$.

DEFINITION 3: *The expression:*

$$\text{APPL}_{i,j}^{A}(\mathbf{q}) \Rightarrow [f_{i,j}^{A}(\mathbf{q}) \leq 0]$$

where:

$$\texttt{APPL}^{A}_{i,j}(\mathbf{q}) = [\vec{\nu}^{\mathcal{A}}_{i}(\mathbf{q}) \cdot (b_{j-1} - b_j) \geq 0] \bigwedge [\vec{\nu}^{\mathcal{A}}_{i}(\mathbf{q}) \cdot (b_{j+1} - b_j) \geq 0]$$

and:

$$f^{A}_{i,j}(\mathbf{q}) = \vec{\nu}^{\mathcal{A}}_{i}(\mathbf{q}) \cdot (b_j - a_i(\mathbf{q}))$$

is called a **C-constraint of type A**. *It is denoted by* $\texttt{CONST}^{A}_{i,j}(\mathbf{q})$.

Type B Contact. In the same manner, a type B contact between a_i and $E^{\mathcal{B}}_{j}$ is feasible for a subrange of orientations of $\mathcal{A}$ determined by two conditions (see Figure 2):

$$(a_{i-1}(\mathbf{q}) - a_i(\mathbf{q})) \cdot \vec{\nu}^{\mathcal{B}}_{j} \geq 0$$

$$(a_{i+1}(\mathbf{q}) - a_i(\mathbf{q})) \cdot \vec{\nu}^{\mathcal{B}}_{j} \geq 0.$$

The conjunction of these two conditions is denoted by $\texttt{APPL}^{B}_{i,j}(\mathbf{q})$ and is called the applicability condition of the type B contact between a_i and $E^{\mathcal{B}}_{j}$. When $\mathbf{q}$ is expressed as $(\mathcal{T}, \Theta)$, it depends only on Θ.

When $\mathcal{A}$ moves in such a way that the type B contact between a_i and $E^{\mathcal{B}}_{j}$ is maintained, the robot's configuration moves in a surface, called a **C-surface** of type B, whose equation is:

$$f^{B}_{i,j}(\mathbf{q}) = 0$$

where:

$$f^{B}_{i,j}(\mathbf{q}) = \vec{\nu}^{\mathcal{B}}_{j} \cdot (a_i(\mathbf{q}) - b_j).$$

$\mathcal{CB}$ lies completely within the closed half-space determined by $f^{B}_{i,j}(\mathbf{q}) \leq 0$.

DEFINITION 4: *The expression:*

$$\texttt{APPL}^{B}_{i,j}(\mathbf{q}) \Rightarrow [f^{B}_{i,j}(\mathbf{q}) \leq 0]$$

where:

$$\texttt{APPL}^{B}_{i,j}(\mathbf{q}) = [(a_{i-1}(\mathbf{q}) - a_i(\mathbf{q})) \cdot \vec{\nu}^{\mathcal{B}}_{j} \geq 0] \bigwedge [(a_{i+1}(\mathbf{q}) - a_i(\mathbf{q})) \cdot \vec{\nu}^{\mathcal{B}}_{j} \geq 0]$$

and:

$$f^{B}_{i,j}(\mathbf{q}) = \vec{\nu}^{\mathcal{B}}_{j} \cdot (a_i(\mathbf{q}) - b_j)$$

is called a **C-constraint of type B**. *It is denoted by* $\texttt{CONST}^{B}_{i,j}(\mathbf{q})$.

TESLA EXHIBIT 1013, Page 111

1.3 Representation of a C-obstacle with C-constraints

The main result presented below is Theorem 1 [Lozano-Pérez, 1983] [Donald, 1984], which describes the C-obstacle $\mathcal{CB}$ as a predicate $\mathtt{CB}(\mathbf{q})$. $\mathtt{CB}(\mathbf{q})$ is constructed as the conjunction of all the C-constraints of types A and B generated by the edges and the vertices of $\mathcal{A}$ and $\mathcal{B}$. We first state a simple lemma that will be used in the proof of Theorem 1.

LEMMA 1: *Two convex polygons do not intersect if and only if one of them contains an edge E such that the other polygon lies entirely in the open outer half-plane of E.*

Proof: If one of the polygons contains an edge E such that the other polygon lies entirely in the open outer half-plane of E, then it is clear that the two polygons do not intersect.

On the other hand, if the two polygons do not intersect, then the shortest distance d between the two polygons is non-zero. Three cases illustrated in Figure 3 are possible:

- The shortest distance d is (uniquely) achieved between a vertex p of a polygon and an open edge E of the other polygon (Figure 3.a). Then the polygon containing p lies entirely in the open outer half-plane of E.

- The shortest distance d is achieved (non-uniquely) between two parallel open edges E and E' (Figure 3.b). Then the polygon containing E' lies in the open outer half-plane of E.

- The shortest distance d is (uniquely) achieved between a vertex p of a polygon — call it $\mathcal{P}$ — and a vertex p' of the other polygon — call it $\mathcal{P}'$ (Figure 3.c). Let E_1 and E_2 (resp. E'_1 and E'_2) be the two edges of $\mathcal{P}$ (resp. $\mathcal{P}'$) abutting p (resp. p') in counterclockwise order. Assume that the supporting line of each of the two edges E_1 and E_2 intersects either one of the two edges E'_1 and E'_2. (If this assumption is not verified, then either E_1 or E_2, or both, satisfy the test of the lemma.) In order for the shortest distance d between $\mathcal{P}$ and $\mathcal{P}'$ to be uniquely achieved between p and p', it is necessary that p lie in a circular arc C of radius d centered at p' (arc in plain line in Figure 3.c); each endpoint of C is a point where the tangent to C is parallel to either E'_1 or E'_2.

TESLA EXHIBIT 1013, Page 112

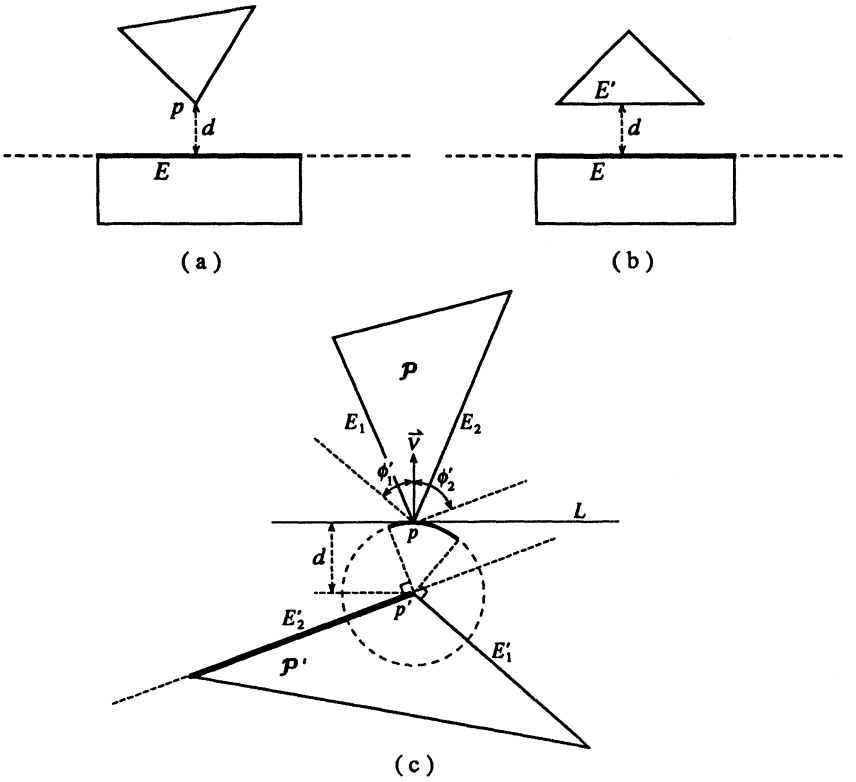


Figure 3. If two convex polygons do not intersect, the shortest distance between them is non-zero and is achieved either between a vertex and an open edge (Figure a), or between two open edges (Figure b), or between two vertices (Figure c). In every case, one polygon contains an edge (shown in bold line) such that the other polygon lies entirely in the open outer half-plane of that edge.

Further, let L be the line tangent to C at p; $\mathcal{P}$ must also lie in one of the two closed half-planes bounded by L, while $\mathcal{P}'$ must lie in the other open half-plane. Let $\overline{L}_{\mathcal{P}}$ denote the closed half-plane containing $\mathcal{P}$ and $\vec{\nu}$ the normal vector to L pointing into $\overline{L}_{\mathcal{P}}$. At p, we erect two half-lines contained in $\overline{L}_{\mathcal{P}}$ and parallel to E_1' and E_2', respectively. Let ϕ_1' and ϕ_2' be the angles made by these half-lines with $\vec{\nu}$. In order for the supporting lines of E_1 and E_2 to intersect either E_1' or E_2', $\mathcal{P}$ must be contained in a cone (not shown in the figure) defined as

follows: its apex is p, its axis points along $\vec{\nu}$, and its sides make the angle $\max\{\phi'_1, \phi'_2\}$ with $\vec{\nu}$. Now, if $\phi'_1 \geq \phi'_2$ then $\mathcal{P}$ lies entirely in the open outer half-plane of E'_1. (This is obvious if p lies in the open outer half-plane of E'_1; on the other hand, if p does not lie in this half-plane, we cannot have $\phi'_1 \geq \phi'_2$.) If $\phi'_2 \geq \phi'_1$, $\mathcal{P}$ lies entirely in the open outer half-plane of E'_2. ∎

THEOREM 1: *Let $\mathcal{A}$ and $\mathcal{B}$ be two convex polygons. The C-obstacle:*

$$\mathcal{CB} = \{\mathbf{q} \in \mathcal{C} \;/\; \mathcal{A}(\mathbf{q}) \cap \mathcal{B} \neq \emptyset\}$$

is such that:

$$\mathbf{q} \in \mathcal{CB} \Leftrightarrow \mathtt{CB}(\mathbf{q})$$

with:

$$\mathtt{CB}(\mathbf{q}) \equiv (\bigwedge_{i,j} \mathtt{CONST}^{A}_{i,j}(\mathbf{q})) \bigwedge (\bigwedge_{i,j} \mathtt{CONST}^{B}_{i,j}(\mathbf{q})).$$

Proof: The expression:

$$\mathbf{q} \in \mathcal{CB} \Leftrightarrow \mathtt{CB}(\mathbf{q})$$

is equivalent to:

$$\mathbf{q} \notin \mathcal{CB} \Leftrightarrow \neg\mathtt{CB}(\mathbf{q})$$

with:

$$\neg\mathtt{CB}(\mathbf{q}) \equiv (\bigvee_{i,j} \neg\mathtt{CONST}^{A}_{i,j}(\mathbf{q})) \bigvee (\bigvee_{i,j} \neg\mathtt{CONST}^{B}_{i,j}(\mathbf{q})).$$

(1) Let $\mathbf{q} \notin \mathcal{CB}$. Hence, $\mathcal{A}(\mathbf{q})$ and $\mathcal{B}$ do not intersect. According to the previous lemma, there exists an edge $E^{\mathcal{A}}_i(\mathbf{q})$ or $E^{\mathcal{B}}_j$ contained in a line L such that $\mathcal{A}(\mathbf{q})$ lies on one side of L and $\mathcal{B}$ on the other side, with either $\mathcal{A}(\mathbf{q})$ or $\mathcal{B}$, but not both, having a non-empty intersection with L. Assume that L contains $E^{\mathcal{A}}_i(\mathbf{q})$ (a similar development could be made if L contained $E^{\mathcal{B}}_j$). Then there exists a vertex b_j of $\mathcal{B}$ such that $\mathtt{APPL}^{A}_{i,j}(\mathbf{q})$ holds. Furthermore, this vertex verifies: $f^{A}_{i,j}(\mathbf{q}) > 0$. Thus, there exists a pair (i,j) such that the C-constraint:

$$\mathtt{CONST}^{A}_{i,j}(\mathbf{q}) \equiv (\mathtt{APPL}^{A}_{i,j}(\mathbf{q}) \Rightarrow f^{A}_{i,j}(\mathbf{q}) \leq 0)$$

does not hold. Therefore:

$$\mathbf{q} \notin \mathcal{CB} \Rightarrow \neg\mathtt{CB}(\mathbf{q}).$$

(2) Let $\neg\mathtt{CB}(\mathbf{q})$ hold. Then there exists at least one pair (i,j) verifying:

$$\mathtt{APPL}^{A}_{i,j}(\mathbf{q}) \quad \text{and} \quad f^{A}_{i,j}(\mathbf{q}) > 0$$

or:

$$\mathtt{APPL}^{B}_{i,j}(\mathbf{q}) \quad \text{and} \quad f^{B}_{i,j}(\mathbf{q}) > 0.$$

In both cases, $\mathcal{A}(\mathbf{q})$ and $\mathcal{B}$ cannot intersect, since they are convex. Hence, $\mathbf{q} \notin \mathcal{CB}$, and:

$$\neg\mathtt{CB}(\mathbf{q}) \Rightarrow \mathbf{q} \notin \mathcal{CB}.$$

■

Using the predicate CB, we can compute whether a configuration $\mathbf{q}$ lies in $\mathcal{CB}$ or not by evaluating every individual C-constraint until one of them evaluates to `false` (then $\mathbf{q} \notin \mathcal{CB}$) or all of them evaluate to `true` (then $\mathbf{q} \in \mathcal{CB}$). The number of C-constraints in $\mathtt{CB}(\mathbf{q})$ (i.e. the worst-case time complexity of this computation) is $2n_{\mathcal{A}}n_{\mathcal{B}}$.

A configuration $\mathbf{q}$ is in $\mathcal{CB}$'s boundary if and only if:

- $\exists i,j$ such that $\mathtt{APPL}^{A}_{i,j}(\mathbf{q})$ (or $\mathtt{APPL}^{B}_{i,j}(\mathbf{q})$) holds, and $f^{A}_{i,j}(\mathbf{q}) = 0$ (or $f^{B}_{i,j}(\mathbf{q}) = 0$),
- $\forall i',j'$ such that $\mathtt{APPL}^{A}_{i',j'}(\mathbf{q})$ holds, $f^{A}_{i',j'}(\mathbf{q}) \leq 0$,
- $\forall i',j'$ such that $\mathtt{APPL}^{B}_{i',j'}(\mathbf{q})$ holds, $f^{B}_{i',j'}(\mathbf{q}) \leq 0$.

If $\mathcal{A}$ and $\mathcal{B}$ are non-convex polygons, they can be represented as finite unions of convex polygons $\mathcal{A}_k$ and $\mathcal{B}_l$:

$$\mathcal{A} = \bigcup_k \mathcal{A}_k \quad \text{and} \quad \mathcal{B} = \bigcup_l \mathcal{B}_l.$$

By defining:

$$\mathcal{CB}_{kl} = \{\mathbf{q} \in \mathcal{C} \;/\; \mathcal{A}_k(\mathbf{q}) \cap \mathcal{B}_l \neq \emptyset\}$$

we get:

$$\mathcal{CB} = \bigcup_{k,l} \mathcal{CB}_{kl}.$$

The predicate $\mathtt{CB}_{kl}$ representing $\mathcal{CB}_{kl}$ can be constructed as in Theorem 1. We have:

$$\mathtt{CB}(\mathbf{q}) \equiv \bigvee_{k,l} \mathtt{CB}_{kl}(\mathbf{q}). \tag{1}$$

The problem of decomposing a non-convex polygon into a union of convex polygons is a classical one in Computational Geometry. Several algorithms have been proposed (e.g. see [Schachter, 1978] [Garey et al., 1978] [Chazelle and Dobkin, 1979 and 1985] [Keil and Sack, 1985] [Chazelle, 1987]). Two types of decomposition can be considered: covering, which does allow the interiors of the convex components to overlap, and partitioning, which disallows the interiors of the convex components from overlapping. Both types of decomposition, including the optimal decomposition into the smallest number of convex polygons, can be performed in polynomial time. In particular, a sweep-line algorithm partitions a non-convex polygon into (non-overlapping) triangles and trapezoids in $O(n \log n)$ time, where n is the number of vertices of the decomposed polygon (see Section 1 of Chapter 5).

When $\mathcal{A}$ is a convex polygon and $\mathcal{B}$ is a regular non-bounded convex polygonal region, we can still use the result of Theorem 1, if we take some precautions. Some vertices b_j are at infinity and do not produce type A contacts. The expressions $\mathtt{CONST}^{A}_{i,j}(\mathbf{q})$ involving these vertices should be modified appropriately, for instance by introducing fictitious vertices at finite distances. If $\mathcal{A}$ is a non-convex polygon and $\mathcal{B}$ is a regular non-bounded non-convex polygonal region, both $\mathcal{A}$ and $\mathcal{B}$ can be decomposed into a possibly non-minimal number of convex parts in polynomial time[1], and a predicate CB can be constructed as in expression (1) to represent $\mathcal{CB}$. It is also not difficult to extend the construction of CB to non-regular polygonal regions.

Since the union of a finite number of polygonal regions is also a polygonal region, we can construct a predicate CB representing the C-obstacle region in $\mathcal{C}$. The free space is then defined by:

$$\mathbf{q} \in \mathcal{C}_{free} \Leftrightarrow \neg \mathtt{CB}(\mathbf{q})$$

[1]The optimal decomposition of a polygon with holes is NP-hard [Lingas, 1982].

TESLA EXHIBIT 1013, Page 116

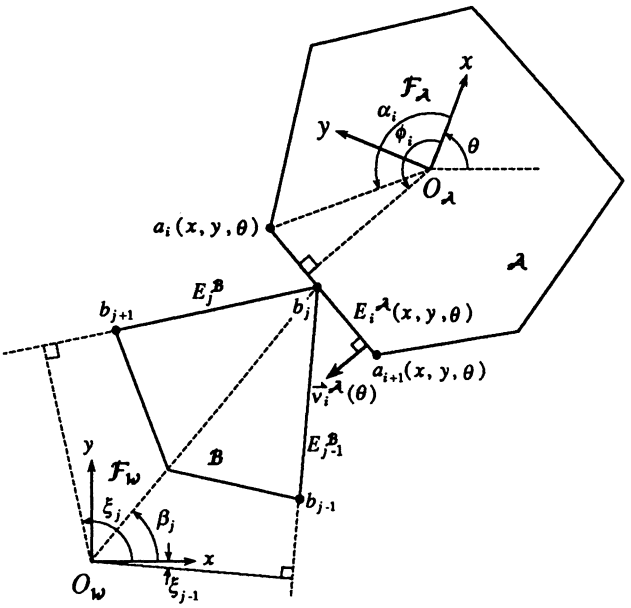


Figure 4. This figure illustrates the notations used in the text for establishing the parameterized form of a C-constraint of type A.

1.4 Parameterization of C-Constraints

In Subsection 1.2 we expressed the C-constraints without assuming any specific parameterization of $\mathcal{C}$. By selecting a representation of the configurations in the form of a list of real parameters, as discussed in Section 4 of Chapter 2, we can derive the corresponding parameterized form of the C-constraints [Brooks and Lozano-Pérez, 1982 and 1983].

For example, let us represent a configuration $\mathbf{q} \in \mathbf{R}^2 \times SO(2)$ by (x, y, θ), where x and $y \in \mathbf{R}$ are the coordinates of $O_\mathcal{A}$ in $\mathcal{F}_\mathcal{W}$, and $\theta \in [0, 2\pi)$ is the angle between the x-axes of $\mathcal{F}_\mathcal{W}$ and $\mathcal{F}_\mathcal{A}$. Figures 4 and 5 illustrate the notations used below.

The outgoing normal vector to $E_i^\mathcal{A}(x, y, \theta)$ is (in $\mathcal{F}_\mathcal{W}$):

$$\vec{v}_i^\mathcal{A}(\theta) = (\cos(\phi_i + \theta), \sin(\phi_i + \theta)).$$

The outgoing normal vector to $E_j^\mathcal{B}$ is:

$$\vec{v}_j^\mathcal{B} = (\cos \xi_j, \sin \xi_j).$$

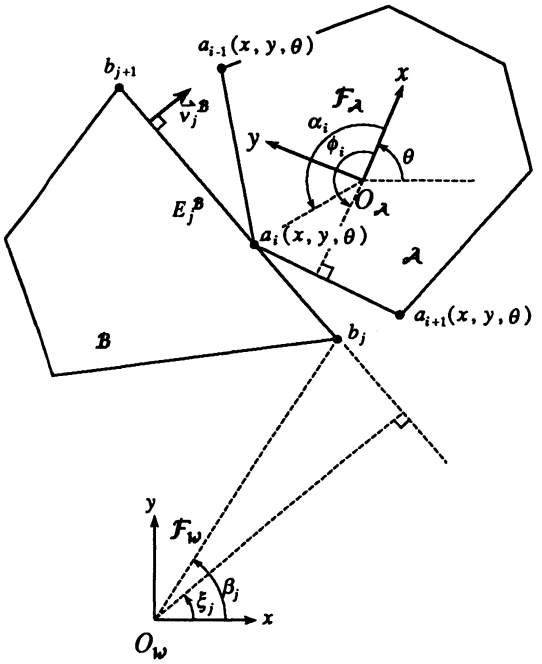


Figure 5. This figure illustrates the notations used in the text for establishing the parameterized form of a C-constraint of type B.

The coordinates of a_i in $\mathcal{F}_A$ are $\|a_i\| \cos \alpha_i$ and $\|a_i\| \sin \alpha_i$, where $\|a_i\|$ is the distance from the origin of $\mathcal{F}_A$ to the vertex a_i. Hence:

$$a_i(x, y, \theta) = (x + \|a_i\| \cos(\alpha_i + \theta), y + \|a_i\| \sin(\alpha_i + \theta)).$$

The vertex b_j is given by:

$$b_j = (\|b_j\| \cos \beta_j, \|b_j\| \sin \beta_j)$$

where $\|b_j\|$ is the distance from the origin of $\mathcal{F}_W$ to b_j.

Consider the type A contact between the edge E_i^A and the vertex b_j shown in Figure 4. The corresponding parameterized C-constraint is:

$$\text{APPL}_{i,j}^A(\theta) \Rightarrow [f_{i,j}^A(x, y, \theta) \leq 0].$$

The applicability condition:

$$\text{APPL}_{i,j}^A(\theta) = [\vec{v}_i^A(\theta) \cdot (b_{j-1} - b_j) \geq 0] \bigwedge [\vec{v}_i^A(\theta) \cdot (b_{j+1} - b_j) \geq 0]$$

TESLA EXHIBIT 1013, Page 118

entails:

$$\theta \in [\xi_{j-1} - \phi_i - \pi, \xi_j - \phi_i - \pi] \pmod{2\pi}.$$

The expression $f^A_{i,j}(x,y,\theta) = \vec{v}^A_i(\theta) \cdot (b_j - a_i(x,y,\theta))$ can be written:

$$-x\cos(\phi_i+\theta) - y\sin(\phi_i+\theta) + \|b_j\|\cos(\phi_i+\theta-\beta_j) - \|a_i\|\cos(\phi_i-\alpha_i).$$

Consider the type B contact between the vertex a_i and the edge $E^{\mathcal{B}}_j$ shown in Figure 5. The corresponding parameterized C-constraint is:

$$\texttt{APPL}^B_{i,j}(\theta) \Rightarrow [f^B_{i,j}(x,y,\theta) \leq 0].$$

The applicability condition:[2]

$$\begin{aligned}\texttt{APPL}^B_{i,j}(\theta) \;=\; & [(a_{i-1}(x,y,\theta) - a_i(x,y,\theta)) \cdot \vec{v}^{\mathcal{B}}_j \geq 0] \\ & \bigwedge [(a_{i+1}(x,y,\theta) - a_i(x,y,\theta)) \cdot \vec{v}^{\mathcal{B}}_j \geq 0].\end{aligned}$$

entails:

$$\theta \in [\xi_j - \phi_i - \pi, \xi_j - \phi_{i-1} - \pi] \pmod{2\pi}.$$

The expression $f^B_{i,j}(x,y,\theta) = \vec{v}^{\mathcal{B}}_j \cdot (a_i(x,y,\theta) - b_j)$ can be written:

$$x\cos\xi_j + y\sin\xi_j + \|a_i\|\cos(\alpha_i+\theta-\xi_j) - \|b_j\|\cos(\beta_j-\xi_j).$$

Using a parameterization of $\mathcal{A}$'s orientation based on the stereographic projection, instead of the angle θ, yields a semi-algebraic expression of $\texttt{CB}(\mathbf{q})$.

1.5 Geometric Interpretation

Let $\mathcal{A}$ and $\mathcal{B}$ be convex polygons, and $\mathbf{q}$ be represented by $(x,y,\theta) \in \mathbf{R}^2 \times [0,2\pi)$ as in the previous subsection. Each C-surface bounding $\mathcal{CB}$ is represented by $f^A_{i,j}(x,y,\theta) = 0$ or $f^B_{i,j}(x,y,\theta) = 0$ in $\mathbf{R}^2 \times [0,2\pi)$. If we keep the orientation of $\mathcal{A}$ constant and equal to θ_0, then both $f^A_{i,j}(x,y,\theta_0) = 0$ and $f^B_{i,j}(x,y,\theta_0) = 0$ represent straight lines in $\mathbf{R}^2$:

- The line $f^A_{i,j}(x,y,\theta_0) = 0$, whose parameterized equation is:

$$-x\cos(\phi_i+\theta_0) - y\sin(\phi_i+\theta_0) + \|b_j\|\cos(\phi_i+\theta_0-\beta_j) - \|a_i\|\cos(\phi_i-\alpha_i) = 0,$$

[2] Despite the appearance, $\texttt{APPL}^B_{i,j}(\theta)$ does not depend on x and y. Indeed, we could as well write:

$\texttt{APPL}^B_{i,j}(\theta) = [(a_{i-1}(0,0,\theta) - a_i(0,0,\theta)) \cdot \vec{v}^{\mathcal{B}}_j \geq 0] \bigwedge [(a_{i+1}(0,0,\theta) - a_i(0,0,\theta)) \cdot \vec{v}^{\mathcal{B}}_j \geq 0]$.

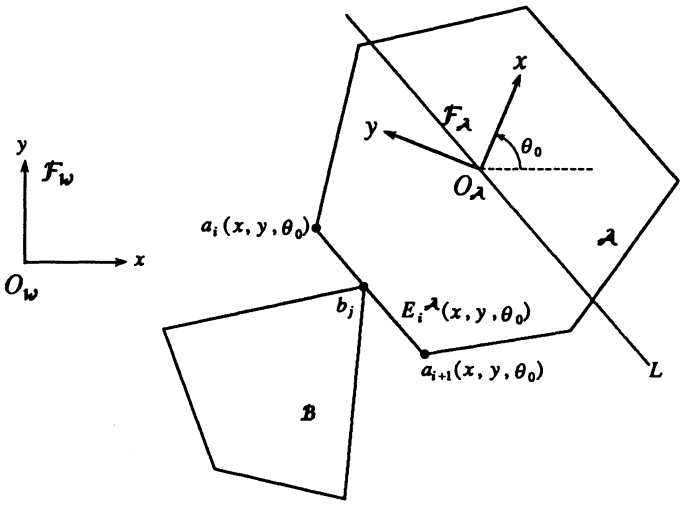


Figure 6. $\mathcal{A}$ is in type A contact with $\mathcal{B}$. If $\mathcal{A}$ translates at a fixed orientation θ_0 while maintaining this contact, the robot's configuration moves along a segment contained in a line L parallel to $E_i^\mathcal{A}(x, y, \theta_0)$. If $\mathcal{A}$ takes a slightly different orientation $\theta_0 + \delta\theta$, the line rotates by the same angle. The vertex b_j is the center of rotation.

is parallel to the edge $E_i^\mathcal{A}(x, y, \theta_0)$. It rotates when $\mathcal{A}$ rotates within the range of orientations where the contact between $E_i^\mathcal{A}$ and b_j is feasible (see Figure 6).

- The line $f_{i,j}^\mathcal{B}(x, y, \theta_0) = 0$, whose equation is:

$$x \cos\xi_j + y \sin\xi_j + \|a_i\| \cos(\alpha_i + \theta_0 - \xi_j) - \|b_j\| \cos(\beta_j - \xi_j) = 0,$$

is parallel to the edge $E_j^\mathcal{B}$. It translates when $\mathcal{A}$ rotates within the range of orientations where the contact between a_i and $E_j^\mathcal{B}$ is feasible (see Figure 7).

Therefore, all the C-surfaces are ruled surfaces (the C-surfaces of type A are helicoids). Figure 8 shows a C-obstacle $\mathcal{CB}$ in $\mathbf{R}^2 \times [0, 2\pi]$. It is a three-dimensional volume bounded by patches of C-surfaces. These patches are called the **faces** of the C-obstacle.

TESLA EXHIBIT 1013, Page 120

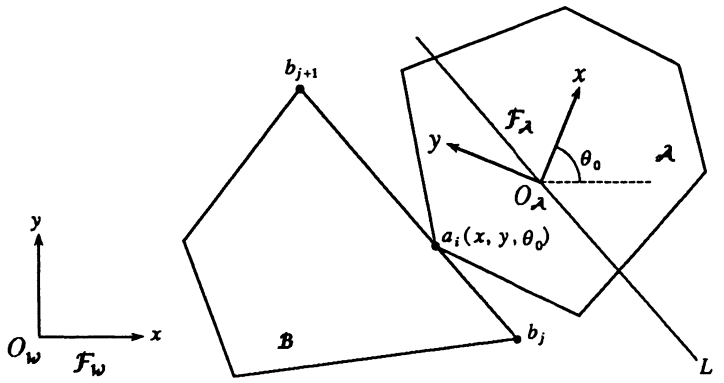


Figure 7. $\mathcal{A}$ is in type B contact with $\mathcal{B}$. If $\mathcal{A}$ translates at a fixed orientation, while maintaining this contact, the robot's configuration moves along a segment contained in a line L parallel to $E_j^{\mathcal{B}}$. If $\mathcal{A}$ takes a slightly different orientation, the line translates accordingly.

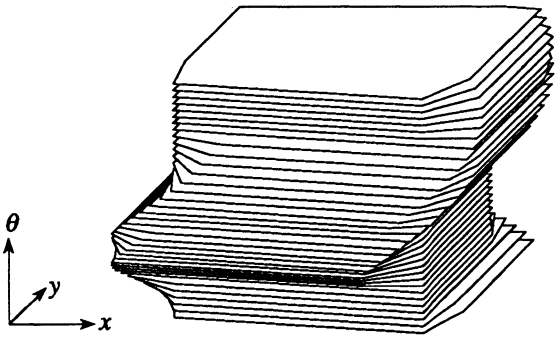


Figure 8. This figure represents a C-obstacle $\mathcal{CB}$ in $\mathbf{R} \times [0, 2\pi]$ when both $\mathcal{A}$ and $\mathcal{B}$ are convex polygons. Each cross-section at a fixed θ is a convex polygon, the C-obstacle in $\mathbf{R}^2$ when $\mathcal{A}$ translates at this fixed orientation. The boundary of $\mathcal{CB}$ is made of patches of ruled surfaces, called faces. Each face corresponds to a type A or type B contact, and is generated either by a rotating line (type A contact) or a translating line (type B contact).

The cross-section $\mathcal{CB}_{\theta_0}$ of $\mathcal{CB}$ at $\theta = \theta_0$ is a region of $\mathbf{R}^2$ defined by:

$$(x, y) \in \mathcal{CB}_{\theta_0} \Leftrightarrow [\bigwedge_{i,j} \mathtt{CONST}^A_{i,j}(x, y, \theta_0)] \bigwedge [\bigwedge_{i,j} \mathtt{CONST}^B_{i,j}(x, y, \theta_0)].$$

$\mathcal{CB}_{\theta_0}$ is the intersection of a finite number of closed half-planes, each bounded by a straight line $f^A_{i,j}(x,y,\theta_0) = 0$ or $f^B_{i,j}(x,y,\theta_0) = 0$. Since $\mathcal{CB}$ is compact and regular (see Propositions 4 and 8 in Chapter 2), $\mathcal{CB}_{\theta_0}$ is a convex polygon equal to the Minkowski difference $\mathcal{B} \ominus \mathcal{A}(0,0,\theta_0)$ (see Proposition 3 in Chapter 2).

We can also remark that:

$$\begin{aligned} |f^A_{i,j}(x,y,\theta_0)| &= |\vec{\nu}^{\mathcal{A}}_i(\theta_0) \cdot (b_j - a_i(x,y,\theta_0))| \\ |f^B_{i,j}(x,y,\theta_0)| &= |\vec{\nu}^{\mathcal{B}}_j \cdot (a_i(x,y,\theta_0) - b_j)| \end{aligned}$$

are the Euclidean distances from the point$(x,y) \in \mathbf{R}^2$ to the lines $f^A_{i,j}(x,y,\theta_0) = 0$ and $f^B_{i,j}(x,y,\theta_0) = 0$, respectively. Thus, when $(x,y,\theta_0) \notin \mathcal{CB}_{\theta_0}$, these values can be used to compute the distance from (x,y,θ_0) to $\mathcal{CB}$ in the subspace $\theta = \theta_0$.

1.6 Boundary Representation (Translational Case)

The predicate CB is conceptually simple. But it does not provide an explicit representation of the boundary of the C-obstacle $\mathcal{CB}$, such as a list of faces, edges and vertices, with both their equations and topological adjacency relation. In this subsection we present a method for constructing such a boundary representation in the case where $\mathcal{A}$ can only translate. In the next subsection, we will describe a more general method that is applicable when $\mathcal{A}$ can both translate and rotate.

Let us first assume that $\mathcal{A}$ and $\mathcal{B}$ are convex polygons and that $\mathcal{A}$ translates at a fixed orientation θ_0. We saw previously that in this case, $\mathcal{CB}_{\theta_0}$ is also a convex polygon. One can easily verify that, if θ_0 is a general orientation of $\mathcal{A}$ relative to $\mathcal{B}$ (i.e. no edge of $\mathcal{A}(x,y,\theta_0)$ is parallel to an edge of $\mathcal{B}$), then the vertices of $\mathcal{CB}_{\theta_0}$ are all the points obtained as follows:

- If $\mathtt{APPL}^A_{i,j}(\theta_0)$ holds, then the points

$$b_j - a_i(0,0,\theta_0) \text{ and } b_j - a_{i+1}(0,0,\theta_0)$$

  are vertices of $\mathcal{CB}_{\theta_0}$.

- If $\mathtt{APPL}^B_{i,j}(\theta_0)$ holds, then the points

$$b_j - a_i(0,0,\theta_0) \text{ and } b_{j+1} - a_i(0,0,\theta_0)$$

  are vertices of $\mathcal{CB}_{\theta_0}$.

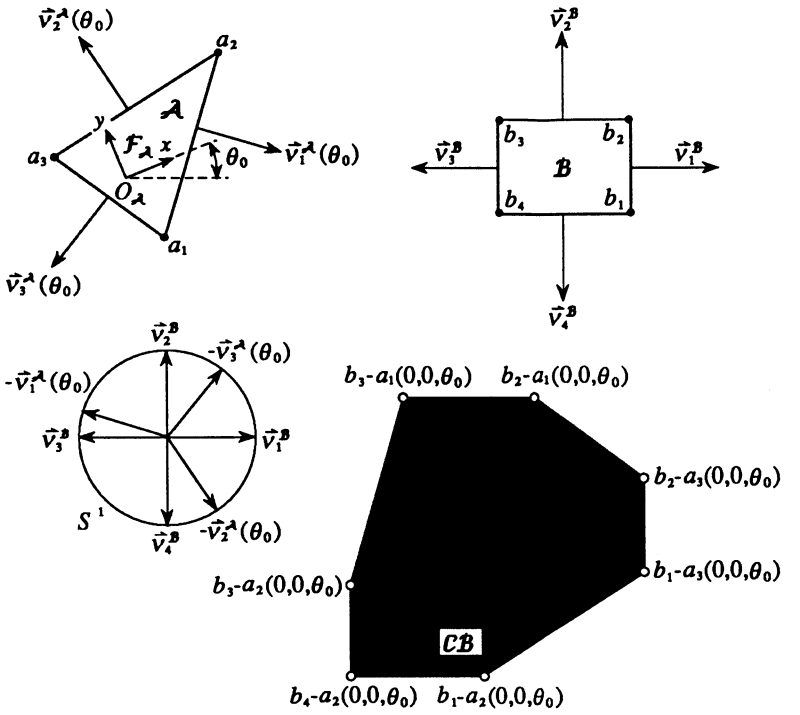


Figure 9. This figure illustrates an efficient algorithm for computing $\mathcal{CB}_{\theta_0}$. The unit vectors $-\vec{\nu}_i^{\mathcal{A}}(\theta_0)$, $i = 1, ..., n_{\mathcal{A}}$, and $\vec{\nu}_j^{\mathcal{B}}$, $j = 1, ..., n_{\mathcal{B}}$ are affixed at the center of a unit circle S^1. By scanning the vectors $-\vec{\nu}_i^{\mathcal{A}}(\theta_0)$ and $\vec{\nu}_j^{\mathcal{B}}$, say in counterclockwise order, one constructs the successive $n_{\mathcal{A}} + n_{\mathcal{B}}$ vertices of $\mathcal{CB}_{\theta_0}$.

This yields the following efficient algorithm for constructing $\mathcal{CB}_{\theta_0}$. This algorithm was originally proposed by Lozano-Pérez [Lozano-Pérez, 1983]. (See also [Guibas, Ramshaw and Stolfi, 1983].)

Let us affix the vectors $-\vec{\nu}_i^{\mathcal{A}}(\theta_0)$, $i = 1, ..., n_{\mathcal{A}}$, and $\vec{\nu}_j^{\mathcal{B}}$, $j = 1, ..., n_{\mathcal{B}}$ at the center of the unit circle S^1 (see Figure 9). Since both $\mathcal{A}$ and $\mathcal{B}$ are convex polygons, the counterclockwise angle between two successive vectors $-\vec{\nu}_i^{\mathcal{A}}(\theta_0)$ and $-\vec{\nu}_{i+1}^{\mathcal{A}}(\theta_0)$, or $\vec{\nu}_j^{\mathcal{B}}$ and $\vec{\nu}_{j+1}^{\mathcal{B}}$, is smaller than π. Therefore, θ_0 being a general orientation of $\mathcal{A}$ relative to $\mathcal{B}$, $\texttt{APPL}_{i,j}^{\mathcal{A}}(\theta_0)$ holds if and only if $-\vec{\nu}_i^{\mathcal{A}}(\theta_0)$ points between $\vec{\nu}_{j-1}^{\mathcal{B}}$ and $\vec{\nu}_j^{\mathcal{B}}$, i.e. $-\vec{\nu}_i^{\mathcal{A}}(\theta_0)$ is a positive linear combination of $\vec{\nu}_{j-1}^{\mathcal{B}}$ and $\vec{\nu}_j^{\mathcal{B}}$. In the same way, $\texttt{APPL}_{i,j}^{\mathcal{B}}(\theta_0)$ holds if and only if $\vec{\nu}_j^{\mathcal{B}}$ points between $-\vec{\nu}_{i-1}^{\mathcal{A}}(\theta_0)$ and $-\vec{\nu}_i^{\mathcal{A}}(\theta_0)$. By scanning the

vectors $-\vec{\nu}_i^{\mathcal{A}}(\theta_0)$ and $\vec{\nu}_j^{\mathcal{B}}$, say in counterclockwise order, one constructs the successive $n_{\mathcal{A}}+n_{\mathcal{B}}$ vertices of $\mathcal{CB}_{\theta_0}$. For instance, if $-\vec{\nu}_i^{\mathcal{A}}(\theta_0)$ is being considered, let $\vec{\nu}_{j-1}^{\mathcal{B}}$ and $\vec{\nu}_j^{\mathcal{B}}$ be the two normals to edges of $\mathcal{B}$ between which it points in S^1. The vertex $b_j - a_{i+1}(0,0,\theta_0)$ is then generated. If $\mathcal{A}$ and $\mathcal{B}$ are given as sequences of vertices specified by their coordinates, the complete boundary of $\mathcal{CB}_{\theta_0}$ can thus be traced out in $O(n_{\mathcal{A}}+n_{\mathcal{B}})$ time, which is optimal.

When $\mathcal{A}$ is at a critical orientation θ_0 such that an edge $E_i^{\mathcal{A}}(x,y,\theta_0)$ of $\mathcal{A}(x,y,\theta_0)$ is parallel to an edge $E_j^{\mathcal{B}}$ of $\mathcal{B}$ and the contact between the two edges is feasible, the points $b_j - a_i(0,0,\theta_0)$, $b_{j+1} - a_i(0,0,\theta_0)$, $b_j - a_{i+1}(0,0,\theta_0)$, and $b_{j+1} - a_{i+1}(0,0,\theta_0)$ are colinear, and the two points $b_{j+1} - a_i(0,0,\theta_0)$ and $b_j - a_{i+1}(0,0,\theta_0)$ are not vertices of $\mathcal{CB}_{\theta_0}$. The situation can easily be recognized (since $-\vec{\nu}_i^{\mathcal{A}}(\theta_0) = \vec{\nu}_j^{\mathcal{B}}$), and the algorithm adapted. The modified algorithm runs in the same time $O(n_{\mathcal{A}}+n_{\mathcal{B}})$.

(Lozano-Pérez [Lozano-Pérez, 1983] proposed another method for computing $\mathcal{CB}_{\theta_0}$ which is slightly less efficient. He proved that:

$$\mathcal{CB}_{\theta_0} = conv(vert(\mathcal{B}) \ominus vert(\mathcal{A}(0,0,\theta_0)))$$

where $vert(\mathcal{P})$ denotes the set of vertices of a polygon $\mathcal{P}$, and $conv(S)$ denotes the convex hull of a set of points S. The computation of the convex hull of a set of points is a well-known problem in Computational Geometry [Preparata and Hong, 1977] [Preparata and Shamos, 1985]. Algorithms have been proposed which run in $O(n \log n)$ time, where n is the number of points in the input set. Here, $n = n_{\mathcal{A}} n_{\mathcal{B}}$.)

If $\mathcal{A}$ and $\mathcal{B}$ are non-convex polygons, $\mathcal{CB}_{\theta_0}$ is still a polygon. Indeed, one can decompose both $\mathcal{A}$ and $\mathcal{B}$ into convex polygons and compute the C-obstacle corresponding to every pair of polygons. The union of these C-obstacles, $\mathcal{CB}_{\theta_0}$, is a polygon. Its boundary is made of edges contained in $O(n_{\mathcal{A}} n_{\mathcal{B}})$ lines. These lines may intersect at $O(n_{\mathcal{A}}^2 n_{\mathcal{B}}^2)$ points. Hence, $\mathcal{CB}_{\theta_0}$ has at most $O(n_{\mathcal{A}}^2 n_{\mathcal{B}}^2)$ vertices. Figure 10 shows an example where the number of vertices of $\mathcal{CB}_{\theta_0}$ is actually proportional to $n_{\mathcal{A}}^2 n_{\mathcal{B}}^2$. The following method computes the representation of $\mathcal{CB}_{\theta_0}$'s boundary in $O(n_{\mathcal{A}}^2 n_{\mathcal{B}}^2 \log n_{\mathcal{A}} n_{\mathcal{B}})$ time [Sharir, 1987]. First, $\mathcal{A}$ (resp. $\mathcal{B}$) is partitioned into $O(n_{\mathcal{A}})$ (resp. $O(n_{\mathcal{B}})$) trapezoids and triangles in $O(n_{\mathcal{A}} \log n_{\mathcal{A}})$ (resp. $O(n_{\mathcal{B}} \log n_{\mathcal{B}})$) time, using a sweep-line technique. By considering the trapezoids and the triangles of the two decompositions pairwise, $\mathcal{CB}_{\theta_0}$ is the union of $O(n_{\mathcal{A}} n_{\mathcal{B}})$ convex polygons, each of

TESLA EXHIBIT 1013, Page 124

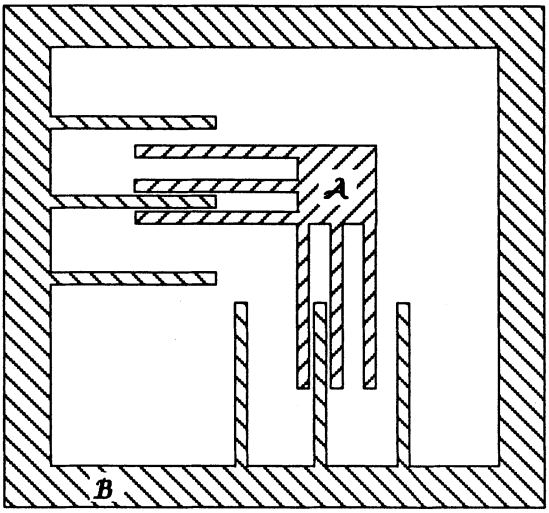


Figure 10. If $\mathcal{A}$ and $\mathcal{B}$ are non-convex polygons, $\mathcal{CB}_{\theta_0}$ is still a polygon. This figure illustrates the case where the number of vertices of $\mathcal{CB}_{\theta_0}$ is proportional to $n_{\mathcal{A}}^2 n_{\mathcal{B}}^2$, where $n_{\mathcal{A}}$ and $n_{\mathcal{B}}$ are the respective numbers of vertices of $\mathcal{A}$ and $\mathcal{B}$.

constant complexity. The boundary of each of these polygons can be computed in constant time (using the method presented above for convex polygons). The $O(n_{\mathcal{A}} n_{\mathcal{B}})$ boundaries can be intersected and $\mathcal{CB}_{\theta_0}$ traced out in $O((n_{\mathcal{A}} n_{\mathcal{B}} + c) \log n_{\mathcal{A}} n_{\mathcal{B}})$ time, with $c \in O(n_{\mathcal{A}}^2 n_{\mathcal{B}}^2)$, using a sweep-line algorithm.

The above method also applies when $\mathcal{B}$ is any regular polygonal region. Hence, when $\mathcal{A}$ can only translate in a polygonal workspace, the boundary of $\mathcal{C}_{free}$ can be computed in $O((n_{\mathcal{A}} n_{\mathcal{B}} + c) \log n_{\mathcal{A}} n_{\mathcal{B}})$ time, where $n_{\mathcal{B}}$ is the total complexity of the workspace obstacles and $c \in O(n_{\mathcal{A}}^2 n_{\mathcal{B}}^2)$. The complexity of this boundary (number of edges and vertices) is $O(n_{\mathcal{A}}^2 n_{\mathcal{B}}^2)$.

1.7 Boundary Representation (General Case)

Let $\mathcal{A}$ and $\mathcal{B}$ be convex polygons. $\mathcal{A}$ can both translate and rotate. The method presented above for computing the boundary of $\mathcal{CB}_{\theta_0}$ in $O(n_{\mathcal{A}} + n_{\mathcal{B}})$ time can be extended to compute the faces, edges and vertices in the boundary of $\mathcal{CB}$, and their adjacency relation. Each face corresponds to the contact of some edge of $\mathcal{A}$ (resp. $\mathcal{B}$) with some vertex of $\mathcal{B}$ (resp.

$\mathcal{A}$). The intersection of the face with a plane at some fixed value of θ is a line segment. The coordinates of its endpoints are functions of θ computable in constant time. These functions define the equations of two of the edges bounding the face. The cross-section through $\mathcal{CB}$ at the same orientation also allows us to compute the adjacency of the face with other faces at this orientation. Some edge equations, the topology of the cross-section, and the adjacency relationships change at $O(n_{\mathcal{A}}n_{\mathcal{B}})$ critical orientations where an edge of $\mathcal{A}$ is aligned with an edge of $\mathcal{B}$. By sweeping a plane at constant θ from $\theta = 0$ to $\theta = 2\pi$ and stopping at every critical orientation, we can construct an explicit representation of the boundary of $\mathcal{CB}$ in time $O(n_{\mathcal{A}}n_{\mathcal{B}}(n_{\mathcal{A}} + n_{\mathcal{B}}))$. This representation has complexity $O(n_{\mathcal{A}}n_{\mathcal{B}}(n_{\mathcal{A}}+n_{\mathcal{B}}))$ (number of faces, edges and vertices). Assuming that $n_{\mathcal{A}} < n_{\mathcal{B}}$, the time and size complexities are $O(n_{\mathcal{A}}n_{\mathcal{B}}^2)$.

If one or both of $\mathcal{A}$ and $\mathcal{B}$ are non-convex, one could think of decomposing them into convex parts, computing the boundaries of the corresponding C-obstacles and intersecting them. However, intersecting boundaries that consist of faces of type A and type B is not easy. Instead, one could try to compute the boundary of the cross-section of the C-obstacle at a fixed orientation of $\mathcal{A}$, and analyze how this boundary varies when $\mathcal{A}$'s orientation changes. But, unlike in the convex case, the cross-section may undergo significant topological changes at orientations other than those where an edge of $\mathcal{A}$ is parallel to an edge of $\mathcal{B}$. Below, we describe another method which directly constructs the boundary of $\mathcal{CB}$ from the description of $\mathcal{A}$ and $\mathcal{B}$. The method is due to Avnaim and Boissonnat [Avnaim and Boissonnat, 1988]. (See also [Koutsou, 1986] and [Brost, 1989] for related work and methods.)

We assume that $\mathcal{A}$ is a polygon and $\mathcal{B}$ is a regular (possibly non-bounded) polygonal region whose edges are all segments of bounded lengths. (If $\mathcal{B}$ is the union of all the obstacles in the workspace, the method described below computes the boundary of $\mathcal{C}_{free}$.) We denote the vertices of $\mathcal{A}$ by a_i, $i = 1, ..., n_{\mathcal{A}}$ and the vertices of $\mathcal{B}$ by b_j, $j = 1, ..., n_{\mathcal{B}}$. We denote the edge of $\mathcal{A}$ connecting a_i to a_{i+1} by $E_i^{\mathcal{A}}$ and the edge of $\mathcal{B}$ connecting b_j to b_{j+1} by $E_j^{\mathcal{B}}$. $\mathcal{B}$'s boundary may contain several closed-loop sequences of edges. In each loop modulo p_k arithmetic is applied to index j, where p_k is the number of vertices in the loop.

We represent a configuration $\mathbf{q}$ as $(x, y, \theta) \in \mathbf{R}^2 \times [0, 2\pi)$, with x and y being the coordinates of $O_{\mathcal{A}}$ in $\mathcal{F}_{\mathcal{W}}$ and θ being the angle between the

TESLA EXHIBIT 1013, Page 126

x-axes of $\mathcal{F}_{\mathcal{W}}$ and $\mathcal{F}_{\mathcal{A}}$. Through modulo 2π arithmetic, $[0, 2\pi)$ is treated as a cyclic interval. As discussed above, the C-obstacle $\mathcal{CB}$ is a regular three-dimensional volume bounded by patches of ruled surfaces (faces). Our goal is to determine these faces, their edges and vertices, and their adjacency relation.

Each face of $\mathcal{CB}$ is either of type A or of type B. In the following, we first examine in detail the faces of type B. After this study, we will consider the faces of type A.

Contact Regions. We define $\mathcal{R}^B_{i,j}$ as the set of all the configurations (x, y, θ) such that $a_i(x, y, \theta) \in E^{\mathcal{B}}_j$ and $int(\mathcal{A}(x, y, \theta)) \cap int(\mathcal{B}) = \emptyset$. $\mathcal{R}^B_{i,j}$ is called a **contact region of type B**. It consists of one or several faces of type B.

As established in Subsection 1.2, $\mathtt{APPL}^B_{i,j}(x, y, \theta)$ is a necessary condition for (x, y, θ) to be in $\mathcal{R}^B_{i,j}$. This condition entails that:

$$\theta \in [\theta^{B,-}_{i,j}, \theta^{B,+}_{i,j}]$$

where $\theta^{B,-}_{i,j} = \xi_j - \phi_i - \pi$ and $\theta^{B,+}_{i,j} = \xi_j - \phi_{i-1} - \pi$ as computed in Subsection 1.4.

Let $\theta_0 \in [\theta^{B,-}_{i,j}, \theta^{B,+}_{i,j}]$. The cross-section of $\mathcal{R}^B_{i,j}$ by the plane $\theta = \theta_0$ is contained in a line segment whose endpoints are $b_j - a_i(0, 0, \theta_0)$ and $b_{j+1} - a_i(0, 0, \theta_0)$. We denote this line segment by $F^B_{i,j}(\theta_0)$.

In order for (x, y, θ_0) to be in $\mathcal{R}^B_{i,j}$, it must also be the case that no open edge of $\mathcal{A}(x, y, \theta_0)$ intersects an open edge of $\mathcal{B}$. Let $P_{k,l}(\theta_0)$ denote the set of configurations (x, y, θ_0) at which the open edge $E^{\mathcal{A}}_k(x, y, \theta_0)$ intersects the open edge $E^{\mathcal{B}}_l$. $P_{k,l}(\theta_0)$ is the interior of a parallelogram whose vertices are: $b_l - a_k(0, 0, \theta_0)$, $b_l - a_{k+1}(0, 0, \theta_0)$, $b_{l+1} - a_{k+1}(0, 0, \theta_0)$, $b_{l+1} - a_k(0, 0, \theta_0)$ (see Figure 11). $P_{k,l}(\theta_0)$ is the intersection of four open half-spaces bounded by four lines with the following respective equations:

$$\begin{aligned} f^A_{k,l}(x, y, \theta_0) &= 0, \\ f^A_{k+1,l}(x, y, \theta_0) &= 0, \\ f^B_{k,l}(x, y, \theta_0) &= 0, \\ f^B_{k,l+1}(x, y, \theta_0) &= 0. \end{aligned}$$

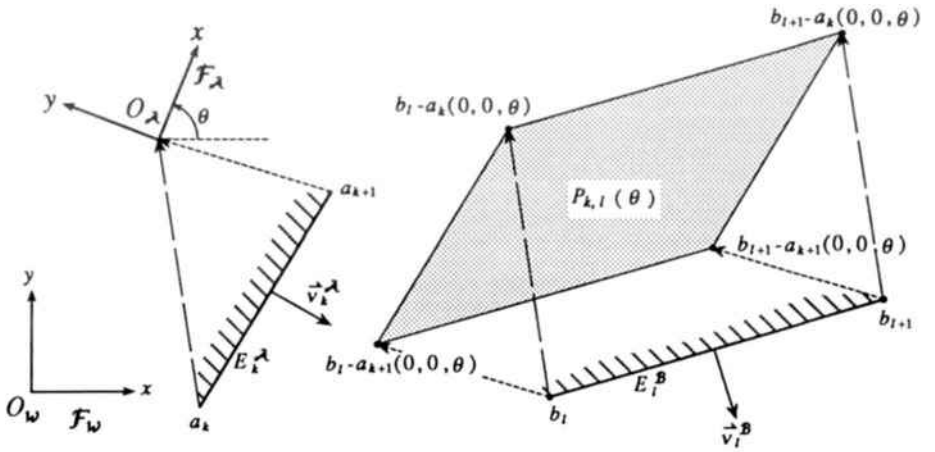


Figure 11. The set of configurations (x, y, θ_0) at which the open edge $E_k^A(x, y, \theta_0)$ intersects the open edge E_l^B is the interior $P_{k,l}(\theta_0)$ of a parallelogram whose vertices are: $b_l - a_k(0, 0, \theta_0)$, $b_l - a_{k+1}(0, 0, \theta_0)$, $b_{l+1} - a_{k+1}(0, 0, \theta_0)$, $b_{l+1} - a_k(0, 0, \theta_0)$.

These equations have been developed in Subsection 1.4. Their forms are:

$$cs(\theta)x + cs(\theta)y + cs(\theta) = 0$$

for the first two, and:

$$\aleph x + \aleph y + cs(\theta) = 0$$

for the last two, where $cs(\theta)$ is an expression of the form:

$$\aleph \cos\theta + \aleph \sin\theta + \aleph$$

and $\aleph$ is a constant. (Each occurrence of $\aleph$ may be a different constant and each occurrence of $cs(\theta)$ may be a different expression.)

We have:

$$\mathcal{R}_{i,j}^B = \bigcup_{\theta \in [\theta_{i,j}^{B,-}, \theta_{i,j}^{B,+}]} \left(F_{i,j}^B(\theta) \setminus \bigcup_{k=1}^{n_A} \bigcup_{l=1}^{n_B} P_{k,l}(\theta) \right).$$

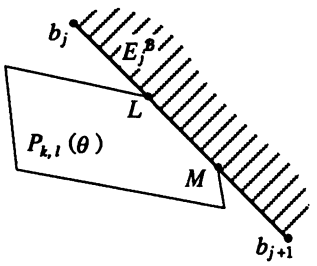


Figure 12. We denote the endpoints of $E_j^{\mathcal{B}} \cap P_{k,l}(\theta)$ by L and M. L is the endpoint which is the closest to b_j.

Intersection of $F_{i,j}^B(\theta)$ with $P_{k,l}(\theta)$. We now compute the intersection of $F_{i,j}^B(\theta)$ with $P_{k,l}(\theta)$ for any orientation $\theta \in [\theta_{i,j}^{B,-}, \theta_{i,j}^{B,+}]$. This intersection is a possibly empty, open segment.

Let us assume for a moment that the origin $O_{\mathcal{A}}$ of the frame $\mathcal{F}_{\mathcal{A}}$ coincides with a_i, so that $F_{i,j}^B(\theta)$ coincides with the edge $E_j^{\mathcal{B}}$, which does not depend on θ. We denote the endpoints of $E_j^{\mathcal{B}} \cap P_{k,l}(\theta)$ by L and M, as illustrated in Figure 12 (L is closer to b_j than M). The coordinates of L and M are of the form $cs(\theta)$.

Let $\lambda_{k,l}^{(i,j)}$ and $\mu_{k,l}^{(i,j)}$ (which we abbreviate in this paragraph by λ and μ) denote the two functions:

$$\lambda_{k,l}^{(i,j)}, \mu_{k,l}^{(i,j)} : [\theta_{i,j}^{B,-}, \theta_{i,j}^{B,+}] \rightarrow [0,1]$$

defined by:

$$\begin{aligned} \lambda_{k,l}^{(i,j)}(\theta) &= \lambda(\theta) = (L - b_j)/(b_{j+1} - b_j), \\ \mu_{k,l}^{(i,j)}(\theta) &= \mu(\theta) = (M - b_j)/(b_{j+1} - b_j). \end{aligned}$$

For every θ where the two functions are defined, we have: $\lambda(\theta) < \mu(\theta)$.

We also know that $\lambda(\theta) = 0$ if and only if $b_j \in cl(P_{k,l}(\theta))$. From the four equations of the lines supporting $P_{k,l}(\theta)$'s edges, we derive four inequalities expressing that $b_j \in cl(P_{k,l}(\theta))$. The inequalities are all of the form $cs(\theta) \leq 0$. The set of values of θ that satisfy a constraint of the form $cs(\theta) \leq 0$ is a single closed connected sub-interval of $[0, 2\pi)$, with 0 and 2π procedurally identified. Therefore, the set of values of θ such

that $\lambda(\theta) = 0$ consists of a finite number of closed disjoint sub-intervals of $[\theta_{i,j}^{B_j-}, \theta_{i,j}^{B_j+}]$ which can be computed in constant time.

In a similar fashion, the set of values of θ such that $\mu(\theta) = 1$ consists of a finite number of closed disjoint sub-intervals of $[\theta_{i,j}^{B_j-}, \theta_{i,j}^{B_j+}]$ which can be computed in constant time.

Let us now assume that $\lambda(\theta) \neq 0$ (resp. $\mu(\theta) \neq 1$). Then L (resp. M) is the intersection of an edge E of $P_{k,l}(\theta)$ with $E_j^{\mathcal{B}}$. Two segments intersect if and only if the two endpoints of both segments are not on the same open side of the line supporting the other segment. Given two points (x_1, y_1) and (x_2, y_2) and a line with equation $f(x, y) = 0$, the two points are not on the same open side of the line if and only if $f(x_1, y_1)f(x_2, y_2) \leq 0$. Therefore, E and $E_j^{\mathcal{B}}$ intersect if and only if two constraints of the form $cs(\theta)cs(\theta) \leq 0$ are satisfied. In order to be satisfied, a constraint of the form $cs(\theta)cs(\theta) \leq 0$ requires either one of two pairs of constraints of the form $cs(\theta) \leq 0$ and $cs(\theta) \geq 0$ to be simultaneously satisfied. Hence, it is satisfied over a finite number of closed disjoint sub-intervals of $[0, 2\pi)$. If E and $E_j^{\mathcal{B}}$ intersect, the coordinates of the intersection point are of the form $cs(\theta)$ if the equation of the line supporting E is of the form $\aleph x + \aleph y + cs(\theta) = 0$, and of the form $cs(\theta)/cs(\theta)$ if the equation of the line supporting E is of the form $cs(\theta)x + cs(\theta)y + cs(\theta) = 0$. While λ is determined by the intersection of the edges of $P_{k,l}(\theta)$ and $E_j^{\mathcal{B}}$, which is the closest to b_j, μ is determined by the intersection that is the closest to b_{j+1}.

Therefore, λ and μ are defined for the same values of θ (those for which $P_{k,l}(\theta)$ intersects $E_j^{\mathcal{B}}$). We can divide the domain of definition of λ and μ, which is a closed subset of $[\theta_{i,j}^{B_j-}, \theta_{i,j}^{B_j+}]$, into a finite number of non-overlapping closed sub-intervals such that in each sub-interval both $\lambda(\theta)$ and $\mu(\theta)$ keep the same analytical expression. The analytical expression of $\lambda(\theta)$ (resp. $\mu(\theta)$) over a sub-interval is one of the following forms:

- 0 (resp. 1),

- $cs(\theta)$,

- $cs(\theta)/cs(\theta)$.

The sub-intervals and the expressions of λ and μ over each of them can be computed in constant time.

TESLA EXHIBIT 1013, Page 130

Computation of $\mathcal{R}^B_{i,j}$. For each parallelogram[3] $P_{k,l}(\theta)$, $k = 1, ..., n_{\mathcal{A}}$, $l = 1, ..., n_{\mathcal{B}}$, we can compute the functions $\lambda^{(i,j)}_{k,l}$ and $\mu^{(i,j)}_{k,l}$ and consider the curves $\xi = \lambda^{(i,j)}_{k,l}(\theta)$ and $\xi = \mu^{(i,j)}_{k,l}(\theta)$ in the rectangle $[0, 2\pi) \times [0, 1]$ over the intervals in which the functions are defined. For each pair (k, l), let $\phi_{k,l}$ denote the region comprised between the curves $\xi = \lambda^{(i,j)}_{k,l}(\theta)$ and $\xi = \mu^{(i,j)}_{k,l}(\theta)$. Let $\Phi^B_{i,j}$ be the region defined as:

$$\Phi^B_{i,j} = [\theta^{B,-}_{i,j}, \theta^{B,+}_{i,j}] \times [0, 1] \setminus \bigcup_{k,l} \phi_{k,l}.$$

Figure 13.a illustrates a region $\Phi^B_{i,j}$. The boundary of $\Phi^B_{i,j}$ corresponds to a double contact between $\mathcal{A}$ and $\mathcal{B}$ (two contacts of type B, or a contact of type B and a contact of type A).

Let $h^B_{i,j}$ be the bijective application defined as follows. For every (θ, ξ) in $[0, 2\pi) \times [0, 1]$, $h^B_{i,j}(\theta, \xi) = (x, y, \theta)$, where x and y are the coordinates of the point $b_j + \xi(b_{j+1} - b_j)$ in $\mathcal{F}_{\mathcal{W}}$. Thus, the image of $\Phi^B_{i,j}$ by $h^B_{i,j}$ is exactly the contact region $\mathcal{R}^B_{i,j}$. In order to compute $\mathcal{R}^B_{i,j}$, we first compute the region $\Phi^B_{i,j}$ and then apply the map $h^B_{i,j}$ to $\Phi^B_{i,j}$.

Let Ψ be a maximal connected subregion of $\Phi^B_{i,j}$ comprised between two orientations θ_1 and θ_2, and between two curves $\xi = \mu^{(i,j)}_{k',l'}(\theta)$ and $\xi = \lambda^{(i,j)}_{k,l}(\theta)$ which keep the same analytical expression in $[\theta_1, \theta_2]$. Ψ is called a **face** of $\Phi^B_{i,j}$ and its mapping in $\mathcal{C}$ by $h^B_{i,j}$ is called a **face** of $\mathcal{CB}$. Such a region Ψ is shown in Figure 13.b.

The faces of the diagram of $\Phi^B_{i,j}$ can be constructed by sweeping a line perpendicular to the θ-axis across the rectangle $[0, 2\pi) \times [0, 1]$. The critical orientations θ_c where the faces may change are all the orientations where:

- a function $\lambda^{(i,j)}_{k,l}$ (and the associated function $\mu^{(i,j)}_{k,l}$) starts/stops being defined,
- the analytical expression of a function $\lambda^{(i,j)}_{k,l}$ or $\mu^{(i,j)}_{k,l}$ changes,
- two functions $\lambda^{(i,j)}_{k,l}$ and $\lambda^{(i,j)}_{k',l'}$, or $\mu^{(i,j)}_{k,l}$ and $\mu^{(i,j)}_{k',l'}$, intersect.

[3] We may discard $P_{i,j}(\theta)$ and $P_{i+1,j}(\theta)$, which cannot intersect $E^{\mathcal{B}}_j$.

TESLA EXHIBIT 1013, Page 131

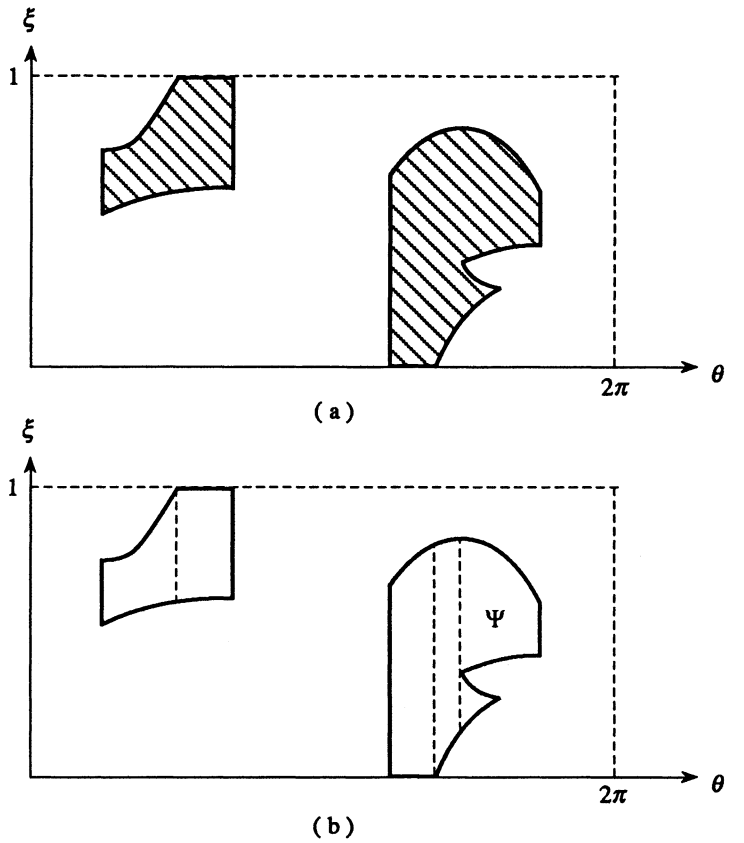


Figure 13. The striped areas in Figure a form a region $\Phi^B_{i,j}$ (see text). Figure b shows the decomposition of this region into six faces. One of them is denoted by Ψ. Each face has at most four edges. The edges contained in the boundary of $\Phi^B_{i,j}$ are called type 2 edges. The other edges (shown in dashed line) are called type 1 edges.

We already know how to compute the critical orientations of the first and second types. The critical orientations of the third types are the intersections of two curves, each having one of the following forms: $\xi = 0$ (or 1), $\xi = cs(\theta)$, and $\xi = cs(\theta)/cs(\theta)$. Every two curves may have up to two intersection points in their common sub-domain of definition. The computation of these points reduces to either the computation of the intersection between a line and a circle, or the computation of the intersection between a circle and a general conic.

TESLA EXHIBIT 1013, Page 132

There are $n_{\mathcal{A}}n_{\mathcal{B}}$ critical orientations of the first and second types. There are $O(n_{\mathcal{A}}^2 n_{\mathcal{B}}^2)$ orientations of the third type. Hence, the sweep-line algorithm computes the faces of $\mathcal{R}_{i,j}^B$ in $O((n_{\mathcal{A}}n_{\mathcal{B}} + c)\log n_{\mathcal{A}}n_{\mathcal{B}})$ time, with $c \in O(n_{\mathcal{A}}^2 n_{\mathcal{B}}^2)$.

Faces, Edges and Vertices of $\mathcal{CB}$. Given a face Ψ of $\Phi_{i,j}^B$, the corresponding face of $\mathcal{CB}$ is obtained by applying the map $h_{i,j}^B$. It is denoted by $h_{i,j}^B(\Psi)$.

The region Ψ is bounded by two curve segments contained in $\xi = \lambda_{k,l}^{(i,j)}(\theta)$ and $\xi = \mu_{k',l'}^{(i,j)}(\theta)$, and possibly by two line segments parallel to the ξ-axis. The application $h_{i,j}^B$ maps these curve segments to curve segments in $\mathbf{R}^2 \times [0, 2\pi)$ which bound the face $h_{i,j}^B(\Psi)$ and form its edges. The intersections of these edges are the vertices of the face.

In order to compute $\Phi_{i,j}^B$, we assumed that a_i and $O_{\mathcal{A}}$ were coincident. If this is not the case, we can compute $\Phi_{i,j}^B$ in the same manner by translating $O_{\mathcal{A}}$ onto a_i; $\mathcal{R}_{i,j}^B$ is then obtained by applying the map $T_{-a_i} \circ h_{i,j}^B$ to $\Phi_{i,j}^B$, where T_{-a_i} denotes the translation of vector $-a_i$. In this way, we can compute the faces of all the type B contact regions $\mathcal{R}_{i,j}^B$, $i = 1, ..., n_{\mathcal{A}}$, $j = 1, ..., n_{\mathcal{B}}$, their edges and their vertices.

Now, we still have to compute the faces of $\mathcal{CB}$ produced by type A contacts. Let:

$$\mathcal{R}_{i,j}^A = \{(x, y, \theta) \in \mathbf{R}^2 \times [0, 2\pi) \;/\; b_j \in E_i^{\mathcal{A}}(x, y, \theta) \;;\; int(\mathcal{A}(x, y, \theta)) \cap int(\mathcal{B}) = \emptyset\}.$$

We can define and compute the region $\Phi_{i,j}^A$ corresponding to $\mathcal{R}_{i,j}^A$ in the same way as $\Phi_{i,j}^B$, by considering that $\mathcal{A}$ is the obstacle and $\mathcal{B}$ is the moving object. This corresponds to taking $\mathbf{q} = (x, y, \theta)$, with x and y being the coordinates of $O_{\mathcal{W}}$ in $\mathcal{F}_{\mathcal{A}}$ and θ being the angle between the x-axis of $\mathcal{F}_{\mathcal{A}}$ and the x-axis of $\mathcal{F}_{\mathcal{W}}$. In order to return to our original parameterization of $\mathcal{C}$, we apply the map INV defined by:

$$INV(x, y, \theta) = (-x\cos\theta - y\sin\theta,\; x\sin\theta - y\cos\theta,\; -\theta).$$

Once the region $\Phi_{i,j}^A$ and its faces have been computed, they are mapped in configuration space by applying $INV \circ T_{-b_j} \circ h_{i,j}^A$.

The computation of all the faces, with their edges and vertices, takes time $O(n_{\mathcal{A}}^3 n_{\mathcal{B}}^3 \log n_{\mathcal{A}} n_{\mathcal{B}})$. The vertices of $\mathcal{CB}$ correspond to triple contacts where the interiors of $\mathcal{A}$ and $\mathcal{B}$ do not intersect. Avnaim and Boissonnat [Avnaim and Boissonnat, 1988] gave an example where the number of such triple contacts is actually proportional to $n_{\mathcal{A}}^3 n_{\mathcal{B}}^3$. Hence, the algorithm is close to optimal.

Adjacency Graph At this point, we have computed all the faces, edges and vertices of $\mathcal{CB}$. It remains to determine their topological adjacency relation. Two faces are adjacent if and only if they share a subset of an edge of non-zero measure in $\mathbf{R}^2$.

We distinguish two types of edges in a face (see Figure 13.b). An edge of type 1 belongs to two different faces of the same contact region $\mathcal{R}$ ($\mathcal{R}_{i,j}^A$ or $\mathcal{R}_{i,j}^B$). An edge of type 2 lies in the boundary of a contact region and thus corresponds to a double contact between $\mathcal{A}$ and $\mathcal{B}$. An edge of type 1 lies in a plane at constant θ; hence, it is always a straight edge. An edge of type 2 may or may not lie in a plane at constant θ; it may or may not be a straight edge.

The adjacency of two faces by an edge of type 1 can be computed during the construction of the Φ regions ($\Phi_{i,j}^A$'s and $\Phi_{i,j}^B$'s) with the sweep-line algorithm, within the same time complexity.

An edge of type 2 corresponds to a double contact that can be identified in constant time during the construction of the Φ regions. During the decomposition of a region Φ into faces, we associate with each double contact the list of the faces of Φ which have an edge of type 2 corresponding to this double contact. By construction, the list is sorted by increasing θ or ξ. Once all the regions have been decomposed into faces, each double contact is labeled by two (possibly empty) lists of faces, which we denote by $\mathcal{L}$ and $\mathcal{L}'$. The double contact may or may not occur at constant θ. If it occurs at a constant θ, say θ_0, $\mathcal{L}$ and $\mathcal{L}'$ are sorted by increasing ξ; a face of $\mathcal{L}$ and a face of $\mathcal{L}'$ are adjacent if their straight edges at θ_0 intersect along a subsegment of non-zero length. If the double contact occurs at non-constant θ, $\mathcal{L}$ and $\mathcal{L}'$ are sorted by increasing θ; a face of $\mathcal{L}$ is adjacent to a face of $\mathcal{L}'$ if and only if the two faces are both defined in an interval of values of θ having non-zero length. In both cases, the adjacency relation among the faces of the two lists can be computed in worst-case time equal to the sum of the lengths of the two

TESLA EXHIBIT 1013, Page 134

lists. Each face has at most four edges of type 2. Hence, the adjacency of faces by edges of type 2 can be computed in time linear in the total number of faces, which is in $O(n_{\mathcal{A}}^3 n_{\mathcal{B}}^3)$.

Thus, the computation of the boundary of $\mathcal{CB}$ takes time $O(n_{\mathcal{A}}^3 n_{\mathcal{B}}^3 \log n_{\mathcal{A}} n_{\mathcal{B}})$ and the resulting representation has size $O(n_{\mathcal{A}}^3 n_{\mathcal{B}}^3)$. Cases where the actual complexity of the method is much smaller are described in [Avnaim and Boissonnat, 1988]. If $\mathcal{A}$ is a line segment and $\mathcal{B}$ contains no three aligned vertices, then the time complexity of the method is $O(n_{\mathcal{B}}^2 \log n_{\mathcal{B}})$. If the number of vertices $\mathcal{A}$ is small, so that it can be considered constant, and if the edges of $\mathcal{B}$ are not concentrated near each other compared to the diameter of $\mathcal{A}$ (a situation of *bounded local complexity* [Sifrony and Sharir, 1986]), then the number of faces of $\mathcal{CB}$ is $O(n_{\mathcal{B}})$ and the method can be adapted to run in time $O(n_{\mathcal{B}} \log n_{\mathcal{B}})$.

2 Polyhedral Workspace

In this section we extend some of the results presented in the previous section to the case of a polyhedral workspace. The C-obstacle $\mathcal{CB}$ corresponding to an obstacle $\mathcal{B}$ is now a six-dimensional region of $\mathcal{C}$ bounded by five-dimensional C-surfaces generated by C-constraints. We will see that C-constraints arise from three types of contact between $\mathcal{A}$ and $\mathcal{B}$:

- **type A contacts**, each between a face of $\mathcal{A}$ and a vertex of $\mathcal{B}$,
- **type B contacts**, each between a vertex of $\mathcal{A}$ and a face of $\mathcal{B}$,
- **type C contacts**, each between an edge of $\mathcal{A}$ and an edge of $\mathcal{B}$.

Each type of contact produces a distinct C-constraint, and, as in the polygonal case, if $\mathcal{A}$ and $\mathcal{B}$ are convex, the predicate CB defined by $\mathbf{q} \in \mathcal{CB} \Leftrightarrow \mathtt{CB}(\mathbf{q})$ can be constructed as the conjunction of all the C-constraints. However, there are usually many more C-constraints in the polyhedral case than in the polygonal one. For instance, if both $\mathcal{A}$ and $\mathcal{B}$ are parallelepipeds, there are 48 different contacts of type A, 48 different contacts of type B, and 144 different contacts of type C!

2.1 Polyhedral Regions and Polyhedra

A plane P in $\mathbf{R}^3$ decomposes the three-dimensional space into two half-spaces. Let $h(x, y, z) = 0$ be the equation of P. The two closed half-

spaces determined by $h(x,y,z) \leq 0$ and $h(x,y,z) \geq 0$ are denoted by $\overline{h}^-$ and $\overline{h}^+$, respectively.

DEFINITION 5: *A* **convex polyhedral region** $\mathcal{P}$ *in* $\mathbf{R}^3$ *is the intersection of a finite number of closed half-spaces. A* **polyhedral region** *is any subset of* $\mathbf{R}^3$ *obtained by taking the union of a finite number of convex polyhedral regions.*

A polyhedral region may not be bounded, nor connected. Its connected components may not be simply-connected.

DEFINITION 6: *A* **polyhedron** *is any polyhedral region that is homeomorphic to the closed unit ball in* $\mathbf{R}^3$.

Hence, a polyhedron is compact, regular, and simply-connected. Its boundary is a surface homeomorphic to the unit sphere.

Let $\mathcal{P}$ be a regular polyhedral region. Each maximal connected planar region in $\mathcal{P}$'s boundary that is not intersected by the rest of $\mathcal{P}$'s boundary, except possibly along its own boundary, is called a **face**. Each face is a polygonal region. Its edges are called the **edges** of $\mathcal{P}$. Every endpoint of an edge is called a **vertex**. An edge is **closed** (resp. **open**) if we include (resp. do not include) its endpoints. A face is **closed** (resp. **open**) if we include (resp. do not include) its bounding edges. If $\mathcal{P}$ is a manifold with boundary, each edge is shared by exactly two faces.

Every face F_i of $\mathcal{P}$ is contained in a plane P_i called the **supporting plane** of F_i. Let $h_i(x,y,z) = 0$ be the equation of P_i. Without loss of generality, we write this equation so that every point in the open face F_i has a neighborhood whose intersection with $\mathcal{P}$ only contains points verifying $h_i(x,y,z) \leq 0$. The half-space $\overline{h_i}^+$ (resp. $\overline{h_i}^-$) is called the **outer** (resp. **inner**) half-space of F_i. The **outgoing** (resp. **ingoing**) normal of F_i is the unit vector normal to F_i and pointing toward the interior of the outer (resp. inner) half-space of F_i.

Let f, e, and v denote respectively the numbers of faces, edges and vertices of a polyhedron. The surface of a polyhedron is isomorphic to a planar subdivision, i.e. the graph whose nodes are the polyhedron's vertices and whose links are the polyhedron's edges is a graph which can

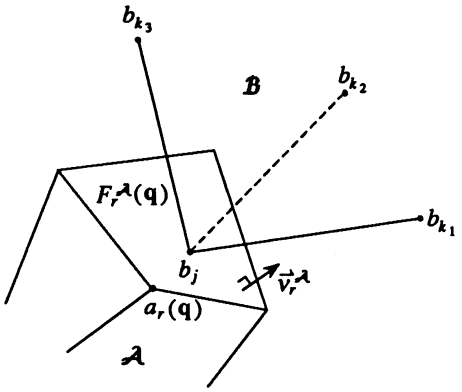


Figure 14. A type A contact between $\mathcal{A}$ and $\mathcal{B}$ occurs when a face $F_r^{\mathcal{A}}$ of $\mathcal{A}$ contains a vertex b_j of $\mathcal{B}$, without overlapping of the interiors of $\mathcal{A}$ and $\mathcal{B}$.

be embedded in the plane without crossings of the links (a planar graph). Hence, the three parameters, f, e and v, are related by Euler's formula:

$$f - e + v = 2.$$

Since each vertex is the extremity of at least three edges and each face is bounded by at least three edges, it can easily be proven that v, e and f are pairwise proportional.

A polyhedron is conveniently represented as a data structure that makes explicit the adjacency relation among its faces, edges, and vertices (e.g. the "doubly-connected-edge-list" (DCEL) [Preparata and Shamos, 1985]). A convex polyhedral region may also be represented by the set of the inequations that defines the half-spaces $\overline{h_i^-}$, i.e. $\bigcap_i \overline{h_i^-}$.

The **complexity** of a polyhedral region is defined as the number of its faces, edges and vertices.

2.2 C-constraints of Type A and Type B

C-constraints of types A and B are established in the same fashion as in the polygonal case. Figures 14 and 15 illustrate the notations used below. The subscript i (resp. j) is used for vertices of $\mathcal{A}$ (resp. $\mathcal{B}$). The subscript l (resp. m) is used for edges of $\mathcal{A}$ (resp. $\mathcal{B}$). The subscript r

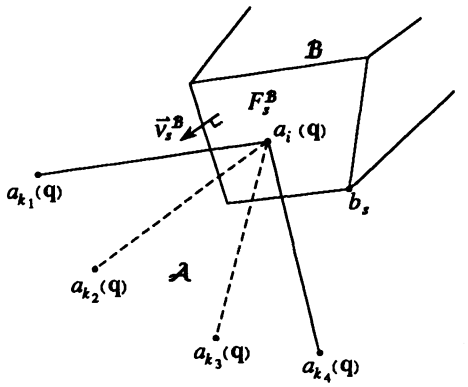


Figure 15. A type B contact between $\mathcal{A}$ and $\mathcal{B}$ occurs when a vertex a_i of $\mathcal{A}$ is contained in a face $F_s^{\mathcal{B}}$ of $\mathcal{B}$, without overlapping of the interiors of $\mathcal{A}$ and $\mathcal{B}$.

(resp. s) is used for faces of $\mathcal{A}$ (resp. $\mathcal{B}$). All edges are treated as closed edges, i.e. they include their endpoints. All faces are treated as closed faces, i.e. they include their bounding edges. In order to simplify the notations, we assume that one extremity of edge $E_l^{\mathcal{A}}$ (resp. $E_m^{\mathcal{B}}$) is a_l (resp. b_m), and that one vertex in the boundary of $\mathcal{A}$'s face $F_r^{\mathcal{A}}$ (resp. $\mathcal{B}$'s face $F_s^{\mathcal{B}}$) is a_r (resp. b_s). We denote the outgoing normal of $F_r^{\mathcal{A}}$ (resp. $F_s^{\mathcal{B}}$) by $\vec{\nu}_r^{\mathcal{A}}$ (resp. $\vec{\nu}_s^{\mathcal{B}}$).

Type A Contact. Consider the type A contact between the face $F_r^{\mathcal{A}}$ of $\mathcal{A}$ and the vertex b_j of $\mathcal{B}$ (Figure 14). The coboundary of b_j is defined as the set of all the edges of $\mathcal{B}$ having b_j as one of their endpoints. We let $\omega(b_j)$ denote the set of all the vertices (except b_j) at the extremities of the edges belonging to the coboundary of b_j. In Figure 14, $\omega(b_j) = \{b_{k_1}, b_{k_2}, b_{k_3}\}$.

DEFINITION 7: *The expression:*

$$\mathtt{APPL}_{r,j}^{A}(\mathbf{q}) \Rightarrow [f_{r,j}^{A}(\mathbf{q}) \leq 0]$$

where:

$$\mathtt{APPL}_{r,j}^{A}(\mathbf{q}) = \bigwedge_{b_k \in \omega(b_j)} [\vec{\nu}_r^{\mathcal{A}}(\mathbf{q}) \cdot (b_k - b_j) \geq 0]$$

and:

$$f_{r,j}^{A}(\mathbf{q}) = \vec{\nu}_r^{\mathcal{A}}(\mathbf{q}) \cdot (b_j - a_r(\mathbf{q}))$$

is called a **C-constraint of type A**. *It is denoted by* $\texttt{CONST}^{A}_{r,j}(\mathbf{q})$.

Type B Contact. Consider the type B contact between the vertex a_i of $\mathcal{A}$ and the face $F^{\mathcal{B}}_s$ of $\mathcal{B}$ (see Figure 15).

DEFINITION 8: *The expression:*

$$\texttt{APPL}^{B}_{i,s}(\mathbf{q}) \Rightarrow [f^{B}_{i,s}(\mathbf{q}) \leq 0]$$

where:

$$\texttt{APPL}^{B}_{i,s}(\mathbf{q}) = \bigwedge_{a_k \in \omega(a_i)} [(a_k(\mathbf{q}) - a_i(\mathbf{q})) \cdot \vec{\nu}^{\mathcal{B}}_s \geq 0]$$

and:

$$f^{B}_{i,s}(\mathbf{q}) = \vec{\nu}^{\mathcal{B}}_s \cdot (a_i(\mathbf{q}) - b_s)$$

is called a **C-constraint of type B**. *It is denoted by* $\texttt{CONST}^{B}_{i,s}(\mathbf{q})$.

The surface (i.e. the five-dimensional submanifold of $\mathcal{C}$) represented by $f^{A}_{r,j}(\mathbf{q}) = 0$ (resp. $f^{B}_{i,s}(\mathbf{q}) = 0$) is called a **C-surface** of type A (resp. type B).

When $\mathbf{q}$ is expressed as $(\mathcal{T}, \Theta)$, the applicability conditions $\texttt{APPL}^{A}_{r,j}(\mathbf{q})$ and $\texttt{APPL}^{B}_{i,s}(\mathbf{q})$ depend only on Θ.

2.3 C-constraints of Type C

A contact of type C occurs when an edge $E^{\mathcal{A}}_l$ of $\mathcal{A}$ is in contact with an edge $E^{\mathcal{B}}_m$ of $\mathcal{B}$, with no intersection of the interiors of the two objects. A type C contact is illustrated in Figure 16.

A C-contact between $E^{\mathcal{A}}_l$ and $E^{\mathcal{B}}_m$ is feasible only for a subrange of orientations of $\mathcal{A}$. This subrange can be determined as follows [Donald, 1984]. Let $P_{l,m}(\mathbf{q})$ be the plane containing both $E^{\mathcal{A}}_l(\mathbf{q})$ and $E^{\mathcal{B}}_m$ when the two edges are in contact (we assume here that $E^{\mathcal{A}}_l(\mathbf{q})$ and $E^{\mathcal{B}}_m$ are not colinear). There is no intersection of the interiors of $\mathcal{A}(\mathbf{q})$ and $\mathcal{B}$ if and only if $P_{l,m}(\mathbf{q})$ is a *separating plane* of $\mathcal{A}(\mathbf{q})$ and $\mathcal{B}$, i.e. if the interiors of $\mathcal{A}(\mathbf{q})$ and $\mathcal{B}$ lie on different sides of $P_{l,m}(\mathbf{q})$.

Let $F^{\mathcal{A}}_{l,1}$ and $F^{\mathcal{A}}_{l,2}$ be the two faces of $\mathcal{A}$ intersecting at $E^{\mathcal{A}}_l$ (see Figure 17.a). Let $F^{\mathcal{B}}_{m,1}$ and $F^{\mathcal{B}}_{m,2}$ be the two faces of $\mathcal{B}$ intersecting at $E^{\mathcal{B}}_m$ (see Figure 17.b). We denote by:

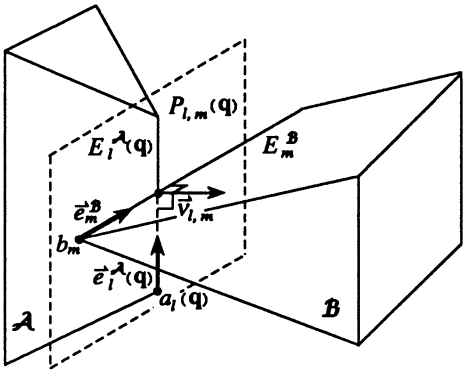


Figure 16. A type C contact between $\mathcal{A}$ and $\mathcal{B}$ occurs when an edge $E_l^{\mathcal{A}}$ of $\mathcal{A}$ intersects an edge $E_m^{\mathcal{B}}$ of $\mathcal{B}$, without overlapping of the interiors of $\mathcal{A}$ and $\mathcal{B}$.

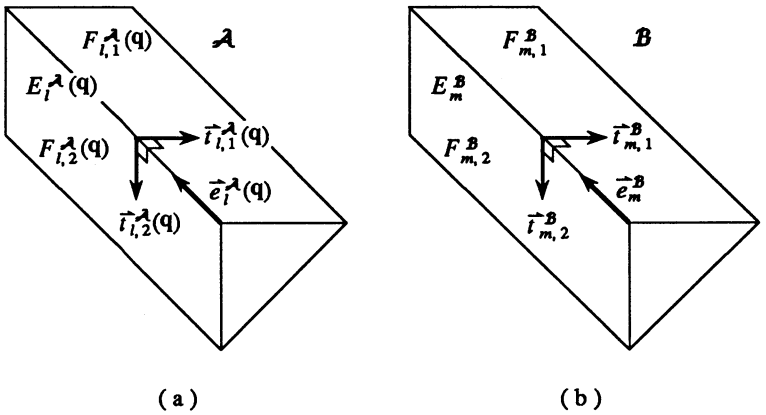


Figure 17. This figure illustrates some of the notations for establishing the general form of a C-constraint of type C (see text).

- $\vec{t}_{l,1}^{\mathcal{A}}(\mathbf{q})$ (resp. $\vec{t}_{l,2}^{\mathcal{A}}(\mathbf{q})$) the vector contained in $F_{l,1}^{\mathcal{A}}(\mathbf{q})$ (resp. $F_{l,2}^{\mathcal{A}}(\mathbf{q})$) and perpendicular to $E_l^{\mathcal{A}}(\mathbf{q})$,
- $\vec{t}_{m,1}^{\mathcal{B}}$ (resp. $\vec{t}_{m,2}^{\mathcal{B}}$) the vector contained in $F_{m,1}^{\mathcal{B}}$ (resp. $F_{m,2}^{\mathcal{B}}$) and perpendicular to $E_m^{\mathcal{B}}$.

Let:

$$\vec{\nu}_{l,m}(\mathbf{q}) = \epsilon_{l,m}\ \vec{e}_l^{\mathcal{A}}(\mathbf{q}) \wedge \vec{e}_m^{\mathcal{B}}$$

TESLA EXHIBIT 1013, Page 140

where:
- $\epsilon_{l,m} = \pm 1$ (the actual value of $\epsilon_{l,m}$ will be defined later),
- $\vec{e}_l^{\mathcal{A}}(\mathbf{q})$ (resp. $\vec{e}_m^{\mathcal{B}}$) denotes a unit vector parallel to $E_l^{\mathcal{A}}(\mathbf{q})$ (resp. $E_m^{\mathcal{B}}$),
- $\vec{u} \wedge \vec{v}$ denotes the outer product of two vectors.

Hence, $\vec{\nu}_{l,m}(\mathbf{q})$ is normal to the plane $P_{l,m}(\mathbf{q})$. Below, we will set $\epsilon_{l,m}$ so that it points toward $\mathcal{B}$.

$P_{l,m}(\mathbf{q})$ is a separating plane of $\mathcal{A}(\mathbf{q})$ and $\mathcal{B}$ if and only if $\vec{t}_{l,1}^{\mathcal{A}}(\mathbf{q})$ and $\vec{t}_{l,2}^{\mathcal{A}}(\mathbf{q})$ point away into one half-space bounded by $P_{l,m}(\mathbf{q})$, while $\vec{t}_{m,1}^{\mathcal{B}}$ and $\vec{t}_{m,2}^{\mathcal{B}}$ point away into the other half-space. This can be expressed by the following three conditions:

$$\begin{array}{lcl} k_{\mathcal{A}} & = & sign(\vec{t}_{l,1}^{\mathcal{A}}(\mathbf{q}) \cdot \vec{\nu}_{l,m}(\mathbf{q})) \quad = \quad sign(\vec{t}_{l,2}^{\mathcal{A}}(\mathbf{q}) \cdot \vec{\nu}_{l,m}(\mathbf{q})), \\ k_{\mathcal{B}} & = & sign(\vec{t}_{m,1}^{\mathcal{B}} \cdot \vec{\nu}_{l,m}(\mathbf{q})) \quad = \quad sign(\vec{t}_{m,2}^{\mathcal{B}} \cdot \vec{\nu}_{l,m}(\mathbf{q})), \\ k_{\mathcal{A}} & \neq & k_{\mathcal{B}}. \end{array}$$

These conditions are called the **applicability conditions** of the type C contact between $E_l^{\mathcal{A}}$ and $E_m^{\mathcal{B}}$. We denote their conjunction by $\mathtt{APPL}_{l,m}^{C}(\mathbf{q})$.

Let $\mathcal{A}$ move in such a way that the type C contact between $E_l^{\mathcal{A}}$ and $E_m^{\mathcal{B}}$ is maintained. During the displacement, the robot's configuration moves in a five-dimensional submanifold of $\mathcal{C}$ whose equation is:

$$f_{l,m}^{C}(\mathbf{q}) = 0$$

where:

$$f_{l,m}^{C}(\mathbf{q}) = \vec{\nu}_{l,m}(\mathbf{q}) \cdot (b_m - a_l(\mathbf{q})).$$

This equation expresses the fact that when $E_l^{\mathcal{A}}(\mathbf{q})$ and $E_m^{\mathcal{B}}$ are in contact, the two vectors $\vec{\nu}_{l,m}(\mathbf{q})$ and $(b_m - a_l(\mathbf{q}))$ are perpendicular or $b_m - a_l(\mathbf{q}) = 0$. The surface it represents is called **C-surface** of type C. It separates $\mathcal{C}$ into two half-spaces. If we take:

$$\epsilon_{l,m} = sign((\vec{e}_l^{\mathcal{A}}(\mathbf{q}) \wedge \vec{e}_m^{\mathcal{B}}) \cdot \vec{t}_{m,1}^{\mathcal{B}})$$

then $\vec{\nu}_{l,m}$ points into the half-space bounded by $P_{l,m}(\mathbf{q})$ which contains $\mathcal{B}$, and the C-obstacle $\mathcal{CB}$ lies completely within the half-space determined by: $f_{l,m}^{C}(\mathbf{q}) \leq 0$.

DEFINITION 9: *Let:*

$$\vec{\nu}_{l,m}(\mathbf{q}) = \epsilon_{l,m}\, \vec{e}_l^{\mathcal{A}}(\mathbf{q}) \wedge \vec{e}_m^{\mathcal{B}}$$

with:

$$\epsilon_{l,m} = sign((\vec{e}_l^{\mathcal{A}}(\mathbf{q}) \wedge \vec{e}_m^{\mathcal{B}}) \cdot \vec{t}_{m,1}^{\mathcal{B}}).$$

The expression:

$$\mathtt{APPL}_{l,m}^{C}(\mathbf{q}) \Rightarrow [f_{l,m}^{C}(\mathbf{q}) \leq 0]$$

where:

$$\begin{aligned} \mathtt{APPL}_{l,m}^{C}(\mathbf{q}) \quad = \quad & [k_{\mathcal{A}} = sign(\vec{t}_{l,1}^{\mathcal{A}}(\mathbf{q}) \cdot \vec{\nu}_{l,m}(\mathbf{q})) = sign(\vec{t}_{l,2}^{\mathcal{A}}(\mathbf{q}) \cdot \vec{\nu}_{l,m}(\mathbf{q}))] \\ & \bigwedge [k_{\mathcal{B}} = sign(\vec{t}_{m,1}^{\mathcal{B}} \cdot \vec{\nu}_{l,m}(\mathbf{q})) = sign(\vec{t}_{m,2}^{\mathcal{B}} \cdot \vec{\nu}_{l,m}(\mathbf{q}))] \\ & \bigwedge k_{\mathcal{A}} \neq k_{\mathcal{B}} \end{aligned}$$

and:

$$f_{l,m}^{C}(\mathbf{q}) = \vec{\nu}_{l,m}(\mathbf{q}) \cdot (b_m - a_l(\mathbf{q}))$$

is called a **C-constraint of type C.** *It is denoted by* $\mathtt{CONST}_{l,m}^{C}(\mathbf{q})$.

In this definition we assume that:

(1) The outer product of $\vec{e}_l^{\mathcal{A}}(\mathbf{q})$ and $\vec{e}_m^{\mathcal{B}}$ is non-zero.

(2) None of the inner products:

$$\vec{t}_{l,1}^{\mathcal{A}}(\mathbf{q}) \cdot \vec{\nu}_{l,m}(\mathbf{q})$$

$$\vec{t}_{l,2}^{\mathcal{A}}(\mathbf{q}) \cdot \vec{\nu}_{l,m}(\mathbf{q})$$

$$\vec{t}_{m,1}^{\mathcal{B}} \cdot \vec{\nu}_{l,m}(\mathbf{q})$$

$$\vec{t}_{m,2}^{\mathcal{B}} \cdot \vec{\nu}_{l,m}(\mathbf{q})$$

is zero.

If condition (1) is not satisfied, it means that $E_l^{\mathcal{A}}(\mathbf{q})$ and $E_m^{\mathcal{B}}$ are colinear. Then, the contact is also a type A and/or type B contact. Hence, one or several C-constraints of type A and/or B already take this situation into account. Similarly, if condition (2) is not satisfied, an edge of $\mathcal{A}$ (or $\mathcal{B}$) is contained in the plane of a face of $\mathcal{B}$ (or $\mathcal{A}$), and again the situation is already taken into account by one or several C-constraints of type A or B. Thus, the situations where the above conditions are not satisfied need not be included in the C-constraints of type C.

2.4 Representation of a C-obstacle with C-constraints

As in the polygonal case, we can express $\mathcal{CB}$ as the conjunction of all the C-constraints of types A, B, and C generated by the faces, vertices, and edges of $\mathcal{A}$ and $\mathcal{B}$ [Donald, 1984].

THEOREM 2: *Let $\mathcal{A}$ and $\mathcal{B}$ be two convex polyhedra. The C-obstacle:*

$$\mathcal{CB} = \{\mathbf{q} \in \mathcal{C} \,/\, \mathcal{A}(\mathbf{q}) \cap \mathcal{B} \neq \emptyset\}$$

is such that:

$$\mathbf{q} \in \mathcal{CB} \Leftrightarrow \mathtt{CB}(\mathbf{q})$$

with:

$$\mathtt{CB}(\mathbf{q}) \equiv (\bigwedge_{r,j} \mathtt{CONST}^{A}_{r,j}(\mathbf{q})) \bigwedge (\bigwedge_{i,s} \mathtt{CONST}^{B}_{i,s}(\mathbf{q})) \bigwedge (\bigwedge_{l,m} \mathtt{CONST}^{C}_{l,m}(\mathbf{q})).$$

Since the numbers of vertices, edges and faces in a polyhedron are piecewise proportional, the total number of C-constraints of Types A, B, and C in $\mathtt{CB}(\mathbf{q})$ (i.e. the time complexity of its evaluation) is proportional to $n_{\mathcal{A}} n_{\mathcal{B}}$, where $n_{\mathcal{A}}$ (resp. $n_{\mathcal{B}}$) is the number of vertices of $\mathcal{A}$ (resp. $\mathcal{B}$). Other forms of the predicate CB have been proposed in [Canny, 1988].

If $\mathcal{A}$ and $\mathcal{B}$ are non-convex polyhedra, then they can be decomposed into convex ones. As in the polygonal case, $\mathtt{CB}(\mathbf{q})$ becomes a disjunction. Every term of the disjunction describes the C-obstacle corresponding to one convex polyhedron resulting from the decomposition of $\mathcal{A}$ and one convex polyhedron resulting from the decomposition of $\mathcal{B}$. The optimal convex decomposition of a non-convex polyhedron is known to be *NP*-hard [Lingas, 1982]. However, a non-optimal decomposition can be produced in polynomial time [Chazelle, 1984] [Bajaj and Dey, 1988]. $\mathtt{CB}(\mathbf{q})$ can also be adapted to handle the case where $\mathcal{B}$ is a non-bounded and/or non-regular polyhedral region.

2.5 Parameterization of C-Constraints

Once a representation of $\mathbf{q}$ has been selected, it can be used to parameterize the C-constraints of types A, B, and C, as we did in the polygonal

case (see Subsection 1.4). However, the resulting expressions are quite long and we do not give them here. In practice, they may be established using an automatic symbolic computation system such as Macsyma (see [Donald, 1984] [Canny, 1988]). If a quaternion-based representation is used, the parameterized expression of $\mathtt{CB}(\mathbf{q})$ can be established in semi-algebraic form.

2.6 Geometric Interpretation

Let us consider the expression of $\mathcal{CB}$ given in Theorem 2 ($\mathcal{A}$ and $\mathcal{B}$ are convex polyhedra), with $\mathbf{q}$ represented as $(\mathcal{T}, \Theta)$, where $\mathcal{T} \in \mathbf{R}^3$ consists of the three coordinates of $O_{\mathcal{A}}$ in $\mathcal{F}_{\mathcal{W}}$ and Θ is any parameterization of the orientation of $\mathcal{A}$. Each C-surface is represented by:

$$f^{A}_{r,j}(\mathcal{T}, \Theta) = 0 \quad \text{or} \quad f^{B}_{i,s}(\mathcal{T}, \Theta) = 0 \quad \text{or} \quad f^{C}_{l,m}(\mathcal{T}, \Theta) = 0.$$

If we fix the orientation of $\mathcal{A}$ equal to Θ_0, then:

$$f^{A}_{r,j}(\mathcal{T}, \Theta_0) = 0, \quad f^{B}_{i,s}(\mathcal{T}, \Theta_0) = 0 \quad \text{and} \quad f^{C}_{l,m}(\mathcal{T}, \Theta_0) = 0$$

represent planes in $\mathbf{R}^3$:

- The plane $f^{A}_{r,j}(\mathcal{T}, \Theta_0) = 0$ is parallel to $F^{\mathcal{A}}_{r}(\mathcal{T}, \Theta_0)$. It rotates if we slightly modify the orientation of $\mathcal{A}$.
- The plane $f^{B}_{i,s}(\mathcal{T}, \Theta_0) = 0$ is parallel to $F^{\mathcal{B}}_{s}$. It translates if we slightly modify the orientation of $\mathcal{A}$.
- The plane $f^{C}_{l,m}(\mathcal{T}, \Theta_0) = 0$ is parallel to both $E^{\mathcal{A}}_{l}(\mathcal{T}, \Theta_0)$ and $E^{\mathcal{B}}_{m}$. It rotates if we slightly modify the orientation of $E^{\mathcal{A}}_{l}(\mathcal{T}, \Theta_0)$.

The cross-section $\mathcal{CB}_{\Theta_0}$ at $\Theta = \Theta_0$ is the intersection of a finite number of closed half-spaces, each bounded by a plane $f^{A}_{r,j}(\mathcal{T}, \Theta_0) = 0$, $f^{B}_{i,s}(\mathcal{T}, \Theta_0) = 0$, or $f^{C}_{l,m}(\mathcal{T}, \Theta_0) = 0$. Since $\mathcal{CB}$ is compact and regular (see Propositions 4 and 8 in Chapter 2), $\mathcal{CB}_{\Theta_0}$ is a convex polyhedron equal to the Minkowski difference $\mathcal{B} \ominus \mathcal{A}(0, \Theta_0)$.

The values of $|f^{A}_{r,j}(\mathcal{T}, \Theta_0)|$, $|f^{B}_{i,s}(\mathcal{T}, \Theta_0)|$, and $|f^{C}_{l,m}(\mathcal{T}, \Theta_0)|$ are the Euclidean distances from $\mathcal{T} \in \mathbf{R}^3$ to the planes $f^{A}_{r,j}(\mathcal{T}, \Theta_0) = 0$, $f^{B}_{i,s}(\mathcal{T}, \Theta_0) = 0$, and $f^{C}_{l,m}(\mathcal{T}, \Theta_0) = 0$, respectively. When $(\mathcal{T}, \Theta_0) \notin \mathcal{CB}_{\Theta_0}$, these values can be used to compute the distance from $(\mathcal{T}, \Theta_0)$ to $\mathcal{CB}$ in the subspace $\Theta = \Theta_0$.

TESLA EXHIBIT 1013, Page 144

2.7 Boundary Representation

Various algorithms have been proposed to compute the explicit representation of the boundary of a C-obstacle $\mathcal{CB}$ when the robot $\mathcal{A}$ can only translate.

If both $\mathcal{A}$ and $\mathcal{B}$ are convex polyhedra, $\mathcal{CB}$ can be expressed as ([Lozano-Pérez, 1983]):

$$\mathcal{CB}_{\Theta_0} = conv(vert(\mathcal{B}) \ominus vert(\mathcal{A}(0, \Theta_0))).$$

There exist algorithms for computing the convex hull of a set of points in three-dimensional space which run in $O(n \log n)$ time, where n is the cardinality of the input set [Preparata and Shamos, 1985]. Here, $n = n_{\mathcal{A}} n_{\mathcal{B}}$.

When $\mathcal{A}$ and $\mathcal{B}$ are convex polyhedra, a more efficient algorithm for constructing $\mathcal{CB}_{\Theta_0}$'s boundary is given in [Guibas and Seidel, 1986]. It runs in $O(n_{\mathcal{A}} + n_{\mathcal{B}} + c)$ time, where c denotes the number of vertices of $\mathcal{CB}_{\Theta_0}$, which is $O(n_{\mathcal{A}} n_{\mathcal{B}})$ in the worst case.

Avnaim and Boissonnat [Avnaim and Boissonnat, 1988] proposed an adaptation of the method described in Subsection 1.7 to compute the boundary of $\mathcal{CB}_{\Theta_0}$ when $\mathcal{A}$ and $\mathcal{B}$ are (possibly non-convex) polyhedra bounded by triangular and/or trapezoidal faces. The adapted method runs in time $O(n_{\mathcal{A}}^3 n_{\mathcal{B}}^3 \log n_{\mathcal{A}} n_{\mathcal{B}})$. The method can be extended with the same asymptotic time complexity to any pair of arbitrary polyhedra by decomposing their faces into triangles and trapezoids. (Each face with n vertices can be decomposed into $O(n)$ triangles and trapezoids in time $O(n \log n)$. Hence, the faces of the two polyhedra can be decomposed into $O(n_{\mathcal{A}})$ and $O(n_{\mathcal{B}})$ components in times $O(n_{\mathcal{A}} \log n_{\mathcal{A}})$ and $O(n_{\mathcal{B}} \log n_{\mathcal{B}})$, respectively.)

3 Semi-Algebraic Workspace

In this section we consider the more general case where both $\mathcal{A}$ and $\mathcal{B}$ are described as arbitrary semi-algebraic subsets of $\mathcal{W} = \mathbf{R}^N$. We show that in this case the C-obstacle $\mathcal{CB}$ can also be represented as a semi-algebraic subset. The polygonal and polyhedral workspaces studied in the previous two sections are particular cases of semi-algebraic workspaces.

3.1 Semi-Algebraic Sets

Semi-algebraic sets are defined by means of first-order logic sentences whose predicates are $=$, $\neq$, $>$, $<$, $\geq$, and $\leq$, whose variables are real numbers and whose terms are multivariate polynomials with rational coefficients. $\mathbf{Q}[X_1; ...; X_n]$ denotes the ring of polynomials in n real variables with rational coefficients.

DEFINITION 10: *An* **atomic polynomial expression** *over* $\mathbf{R}^n$ *is one of the form:*

$$P(x) \bowtie 0$$

where $P \in \mathbf{Q}[X_1; ...; X_n]$, $x \in \mathbf{R}^n$, *and* $\bowtie$ *can be any comparator in* $\{=, \neq, >, <, \geq, \leq\}$. *A* **polynomial expression** *over* $\mathbf{R}^n$ *is any finite boolean combination of atomic polynomial expressions over* $\mathbf{R}^n$. *A subset of* $\mathbf{R}^n$ *whose points satisfy a polynomial expression is called a* **semi-algebraic set.**

DEFINITION 11: *A* **Tarski sentence** *is any polynomial expression prefixed by a finite number (possibly zero) of* $\exists$ *and* $\forall$ *quantifiers applying to some of the variables in the sentence over the set of the reals.*

For example:

$$\forall x,\ \exists y : (x^3 + y^3 > 0) \Rightarrow (x^2 + y^2 + z \leq 0)$$

is a Tarski sentence. A polynomial expression is a quantifier-free Tarski sentence.

The variables in a Tarski sentence which are not quantified (e.g. z in the above example) are called **free** variables. A Tarski sentence without free variables is either `true` or `false`. The **first-order theory of the real numbers**, denoted by $\mathtt{Th}(\mathbf{R})$, is the set of all the Tarski sentences without free variables which are `true`. For example:

$$\exists x : x^2 - 1 = 0$$

belongs to $\mathtt{Th}(\mathbf{R})$, but:

$$\exists x : x^2 + 1 \leq 0$$

does not.

Let $\phi(x_1, ..., x_n)$ denote a Tarski sentence in which all occurrences of the variables $x_1, ..., x_n$ are free, each x_i, $i \in [1, n]$, may or may not occur, and there are no free variables other than $x_1, ..., x_n$. $\phi(x_1, ..., x_n)$ is **satisfiable** if and only if the sentence $\exists x_1, ...\exists x_n : \phi(x_1, ..., x_n)$ is in $\mathrm{Th}(\mathbf{R})$. If it is satisfiable, $\phi(x_1, ..., x_n)$ defines a non-empty subset of $\mathbf{R}^n$. The points of this subset are all the lists $(a_1, ..., a_n)$ of constant real numbers such that $\phi(a_1, ..., a_n)$ is `true`.

Two Tarski sentences $\phi_1(x_1, ..., x_n)$ and $\phi_2(x_1, ..., x_n)$ are said to be **equivalent** if for all real numbers $a_1, ..., a_n$, $\phi_1(a_1, ..., a_n)$ is `true` if and only if $\phi_2(a_1, ..., a_n)$ is `true`, i.e. if the sentence:

$$\forall x_1, \ ... \ , \forall x_n : \phi_1(x_1, ..., x_n) \Leftrightarrow \phi_2(x_1, ..., x_n)$$

belongs to $\mathrm{Th}(\mathbf{R})$.

Tarski showed that any Tarski sentence is equivalent to a (quantifier-free) polynomial expression [Tarski, 1951]. For instance, if $P(x_1, ..., x_n, x_{n+1})$ is a polynomial expression over $\mathbf{R}^{n+1}$, there exist two polynomial expressions over $\mathbf{R}^n$, $Q(x_1, ..., x_n)$ and $R(x_1, ..., x_n)$, such that $Q(x_1, ..., x_n)$ is equivalent to $\exists x_{m+1} : P(x_1, ..., x_m, x_{m+1})$ and $R(x_1, ..., x_n)$ is equivalent to $\forall x_{m+1} : P(x_1, ..., x_m, x_{m+1})$. Several constructive proofs of this result have been given, including one by Tarski. In particular, Collins proposed a quantifier-elimination algorithm (known as the Collins decomposition; see Section 3 of Chapter 5) that takes double-exponential time in the number of variables [Collins, 1975]. Grigoryev [Grigoryev, 1988] described another method that takes simple-exponential time in the number of variables and double-exponential time only in the number of quantifier alternations. By using these algorithms, one can reduce any Tarski sentence containing no free variable to either $0 = 0$ (if the sentence is `true`) or $1 = 0$ (if the sentence is `false`), and thus compute whether the value of the sentence is `true` or `false` (hence, the first-order theory of the real numbers is decidable). One can also check whether a Tarski sentence containing free variables is satisfiable by existentially quantifying the free variables and eliminating them. In addition, if a Tarski sentence is `true` or satisfiable, the methods of Collins and Grigoryev return a sample list of values for the free and existentially quantified variables which satisfy the sentence (see Chapter 5). Various applications of quantifier elimination are presented in [Arnon, 1988].

TESLA EXHIBIT 1013, Page 147

Hence, any Tarski sentence $\phi(x_1, ..., x_n)$ containing n free variables defines a semi-algebraic subset X of $\mathbf{R}^n$. $\phi(x_1, ..., x_n)$ is called the **defining formula** of X.

Given two semi-algebraic subsets of $\mathbf{R}^n$, A and B, with defining formulae $\phi_A(x_1, ..., x_n)$ and $\phi_B(x_1, ..., x_n)$, then the sets $A \cup B$ and $A \cap B$ are semi-algebraic sets with defining formulae $\phi_A(x_1, ..., x_n) \vee \phi_B(x_1, ..., x_n)$ and $\phi_A(x_1, ..., x_n) \wedge \phi_B(x_1, ..., x_n)$. The expression $A \subseteq B$ is a logical expression that translates into the Tarski sentence $\forall x_1, ..., \forall x_n : \phi_A(x_1, ..., x_n) \Rightarrow \phi_B(x_1, ..., x_n)$. In a similar way, $(x_1, ..., x_n) \in A$ translates to the sentence $\phi_A(x_1, ..., x_n)$. These simple rewriting rules will allow us to construct semi-algebraic sets without explicitly giving all their defining formulae, and conversely, will allow us to write Tarski sentences using semi-algebraic sets.

3.2 C-Obstacles

Let every object in $\mathcal{W} = \mathbf{R}^N$ be represented as a semi-algebraic set. We denote the defining formulae of $\mathcal{A}$ and $\mathcal{B}$ by $\Lambda_{\mathcal{A}}(a)$ and $\Lambda_{\mathcal{B}}(b)$, respectively, with a and $b \in \mathbf{R}^N$. $\Lambda_{\mathcal{A}}(a)$ represents $\mathcal{A}$ at the reference configuration.

The C-obstacle $\mathcal{CB} = \{\mathbf{q} \in \mathcal{C} \,/\, \mathcal{A}(\mathbf{q}) \cap \mathcal{B} \neq \emptyset\}$ can be described as:

$$\mathcal{CB} = \{\mathbf{q} \in \mathcal{C} \,/\, \exists a,\ \exists b : \Lambda_{\mathcal{A}}(a) \wedge \Lambda_{\mathcal{B}}(b) \wedge (a(\mathbf{q}) = b)\}.$$

As shown in Section 2 of Chapter 2, any configuration $\mathbf{q}$ of $\mathcal{A}$ can be represented by a vector $\mathbf{u} = (\mathcal{T}, \Theta)$ of $N(N+1)$ real numbers related by $\frac{1}{2}N(N+1)+1$ polynomial equations with integer coefficients. These equations express the fact that the $N \times N$ matrix Θ representing the orientation of $\mathcal{A}$ has orthonormal columns and determinant $+1$. Let $\Gamma(\mathbf{u})$ denote the conjunction of these equations. Let $\Upsilon(\mathbf{u}, a, b)$, with $a \in \mathbf{R}^N$ and $b \in \mathbf{R}^N$, be the conjunction of the N polynomial equations expressing that the coordinates of a point a in $\mathcal{A}$ equal those of a point b in $\mathcal{B}$ when $\mathcal{A}$ is at a configuration represented by $\mathbf{u} = (\mathcal{T}, \Theta)$. These equations are derived from the relation $b = \Theta\, a\, +\, \mathcal{T}$.

Thus $\mathcal{CB}$ can be represented in $\mathbf{R}^{N(N+1)}$ as:

$$\mathcal{CB} = \{\mathbf{u} \in \mathbf{R}^{N(N+1)} \,/\, \mathtt{CB}(\mathbf{u})\}$$

TESLA EXHIBIT 1013, Page 148

with:

$$\mathtt{CB}(\mathbf{u}) \equiv \exists a,\ \exists b : \Lambda_{\mathcal{A}}(a) \wedge \Lambda_{\mathcal{B}}(b) \wedge \Upsilon(\mathbf{u}, a, b) \wedge \Gamma(\mathbf{u}).$$

The expression $\exists a,\ \exists b : \Lambda_{\mathcal{A}}(a) \wedge \Lambda_{\mathcal{B}}(b) \wedge \Upsilon(\mathbf{u}, a, b) \wedge \Gamma(\mathbf{u})$ is a Tarski sentence. Hence, $\mathcal{CB}$ is a semi-algebraic subset of $\mathbf{R}^{N(N+1)}$.

PROPOSITION 1: *If the robot and all the obstacles (in finite number) are represented as semi-algebraic subsets of* $\mathbf{R}^N$*, then every C-obstacle is a semi-algebraic subset of* $\mathbf{R}^{N(N+1)}$*.* $\mathcal{C}_{free}$*,* $\mathcal{C}_{contact}$*, and* $\mathcal{C}_{valid}$ *are also semi-algebraic subsets of* $\mathbf{R}^{N(N+1)}$*.*

The same result can be obtained in $\mathbf{R}^m$, with $m = \frac{1}{2}N(N+1)$, rather than in $\mathbf{R}^{N(N+1)}$, by using an appropriate non-redundant parameterization (chart) of $\mathcal{C}$, e.g. the stereographic projection in the two-dimensional case and unit quaternions in the three-dimensional case (see Section 4 of Chapter 2). Let $\mathbf{x}$ denote the representation of $\mathbf{q}$ in such a chart. The relation $a(\mathbf{q}) = b$ entails a polynomial expression $\Upsilon'(\mathbf{x}, a, b)$. We have:

$$\mathcal{CB} = \{\mathbf{x} \in \mathbf{R}^m \ /\ \mathtt{CB}(\mathbf{x})\}$$

with:

$$\mathtt{CB}(\mathbf{x}) \equiv \exists a,\ \exists b : \Lambda_{\mathcal{A}}(a) \wedge \Lambda_{\mathcal{B}}(b) \wedge \Upsilon'(\mathbf{x}, a, b).$$

Such a parameterization significantly reduces the number of variables in the semi-algebraic representation of $\mathcal{CB}$, and hence the cost of applying quantifier-elimination procedures. However, some extra-work may be needed in order to switch from one chart to another.

Exercises

1: $\mathcal{A}$ and $\mathcal{B}$ are two convex polygons. $\mathcal{A}$ can translate and rotate freely in the plane. Give a computable expression of the predicate $\partial\mathtt{CB}$ representing the boundary of $\mathcal{CB}$, i.e. :

$$\mathbf{q} \in \partial\mathcal{CB} \Leftrightarrow \partial\mathtt{CB}(\mathbf{q}).$$

Extend this result to the case where $\mathcal{A}$ and $\mathcal{B}$ are non-convex polygons.

TESLA EXHIBIT 1013, Page 149

2: Describe a sweep-line algorithm that partitions a non-convex polygon into triangles and trapezoids. [Hint: See Section 1 of Chapter 5.]

3: Using the representation of $\mathcal{CB}$ given in Theorem 1, propose a method for computing the intersection of a pure rotational path with the boundary of a C-obstacle $\mathcal{CB}$ [Donald, 1984]. (Assume that $\mathcal{A}$ and $\mathcal{B}$ are convex polygons.)

4: Let $\mathcal{A}$ and $\mathcal{B}$ be convex polygons. A configuration of $\mathcal{A}$ is parameterized by (x, y, θ), with x and y being the coordinates of $O_{\mathcal{A}}$ in $\mathcal{F}_{\mathcal{W}}$ and θ being the angle between the x-axes of $\mathcal{F}_{\mathcal{W}}$ and $\mathcal{F}_{\mathcal{A}}$. The C-obstacle $\mathcal{CB}$ is a three-dimensional volume in $\mathbf{R}^2 \times [0, 2\pi)$ bounded by patches of C-surfaces of type A and type B (faces). Draw the projection of a face of type A (resp. type B) on the xy-plane. Specify the parameters of the figure (lengths of straight segments, radii of circular arcs). [Hint: See Subsection 3.2 of Chapter 6.]

5: Let $\mathcal{A}$ and $\mathcal{B}$ be two line segments of bounded lengths. Describe the C-obstacle $\mathcal{CB}$ in $\mathbf{R}^2 \times [0, 2\pi)$. How many faces does it have?

6: Let $\mathcal{A}$ and $\mathcal{B}$ be two convex polygons. $\mathcal{A}$ translates at a fixed orientation θ_0. Show that:

$$\mathcal{CB}_{\theta_0} = conv(vert(\mathcal{B}) \ominus vert(\mathcal{A}(0, 0, \theta_0)))$$

where $vert(\mathcal{P})$ denotes the set of vertices of a polygon$\mathcal{P}$ and $conv(S)$ denotes the convex hull of a planar set of points S. Show that the result still holds in the case where $\mathcal{A}$ and $\mathcal{B}$ are convex polyhedra, with $conv(S)$ denoting the convex hull of a set of points in $\mathbf{R}^3$ [Lozano-Pérez, 1983].

7: Let $\mathcal{A}$ and $\mathcal{B}$ be two convex polygons.

a. Let us first assume that $\mathcal{A}$ can only translate at a fixed orientation θ_0. Write a program that computes the boundary of the C-obstacle $\mathcal{CB}_{\theta_0}$ using the method described in Subsection 1.6.

b. Let us now consider the case where $\mathcal{A}$ can translate and rotate freely. Extend the previous program in order to compute the boundary of $\mathcal{CB}$ using the method sketched at the beginning of Subsection 1.7.

TESLA EXHIBIT 1013, Page 150

8: A *generalized polygon* is a region of $\mathbf{R}^2$ homeomorphic to the closed unit disc, whose boundary (a Jordan curve) is a closed-loop sequence of straight segments and circular arcs. Let $\mathcal{A}$ and $\mathcal{B}$ be two convex generalized polygons. Show that the C-obstacle $\mathcal{CB}$ is also a convex generalized polygon.

9: Explain how Figure 10 illustrates the fact that the complexity of $\mathcal{CB}_{\theta_0}$ can be proportional to $n_{\mathcal{A}}^2 n_{\mathcal{B}}^2$.

10: Show that the computation of the intersection of two curves $\xi = \aleph$ and $\xi = cs(\theta)$, where $cs(\theta)$ is an expression of the form $\aleph \cos\theta + \aleph \sin\theta + \aleph$ and $\aleph$ is a constant (each occurrence of $\aleph$ may be a different constant and each occurrence of $cs(\theta)$ may be a different expression), can be reduced to the computation of the intersection of a line and a circle. Show that the computation of the intersection of two curves of the form $\xi = cs(\theta)$ and/or $\xi = cs(\theta)/cs(\theta)$ can be reduced to the computation of a circle and a general conic (see Subsection 1.7).

11: Let $\mathcal{A}$ be a line segment of bounded length. $\mathcal{B}$ is a regular (possibly non-bounded) polygonal region whose edges are all segments of bounded lengths. A contact configuration between $\mathcal{A}$ and $\mathcal{B}$ is a configuration $\mathbf{q}$ for which there exists a pair (X, E) such that X is a vertex of $\mathcal{B}$ (resp. an endpoint of $\mathcal{A}(\mathbf{q})$), E is $\mathcal{A}(\mathbf{q})$ (resp. an edge of $\mathcal{B}$), and X is contained in E. A triple-contact configuration between $\mathcal{A}$ and $\mathcal{B}$ is a configuration $\mathbf{q}$ such that there exist three distinct such pairs (X_i, E_i), $i = 1, 2, 3$. A (triple-)contact configuration $\mathbf{q}$ is semi-free if and only if $\mathcal{A}(\mathbf{q})$ does not intersect the interior of $\mathcal{B}$.

Let $n_{\mathcal{B}}$ be the number of vertices and edges of $\mathcal{B}$. Assume that no three vertices of $\mathcal{B}$ are aligned. Analyze the various types of triple-contact configurations. Show that there are $O(n_{\mathcal{B}} \log n_{\mathcal{B}})$ semi-free triple-contact configurations of $\mathcal{A}$. Derive from these results the fact that the method of Avnaim and Boissonnat (Subsection 1.7) can compute the boundary of $\mathcal{CB}$ in $O(n_{\mathcal{B}}^2 \log n_{\mathcal{B}})$ time and that the size of the boundary is $O(n_{\mathcal{B}}^2)$ [Avnaim and Boissonnat, 1988].

12: Let f, e and v be the number of faces, edges and vertices in a polyhedron. Show that f, e and v are pairwise proportional.

TESLA EXHIBIT 1013, Page 151

13: Prove Theorem 2.

14: Let $\mathcal{A}$ and $\mathcal{B}$ be two convex polyhedra. We call the *wedge* of an edge E (of $\mathcal{A}$ or $\mathcal{B}$) the intersection of the two (closed) inner half-spaces of the two faces intersecting at E. Show that $\mathcal{A}$ and $\mathcal{B}$ have a null intersection if and only if all edges of $\mathcal{A}$ (or $\mathcal{B}$) are outside some wedge of an edge of $\mathcal{B}$ (or $\mathcal{A}$). From this result, derive an expression of the predicate CB defined by $\mathbf{q} \in \mathtt{CB} \Leftrightarrow \mathtt{CB}(\mathbf{q})$ [Canny, 1988].

15: Describe how the method of Subsection 1.7 can be adapted to compute the boundary of a C-obstacle when $\mathcal{A}$ is a polyhedron that can only translate and $\mathcal{B}$ is a polyhedron. $\mathcal{A}$ and $\mathcal{B}$ may not be convex. What is the complexity of the adapted method [Avnaim and Boissonnat, 1988]?

16: Let A be a semi-algebraic subset of $\mathbf{R}^n$ with $\phi_A(x_1, ..., x_n)$ as its defining formula. Show that the interior, the boundary and the closure of A are all semi-algebraic subsets of $\mathbf{R}^n$.

TESLA EXHIBIT 1013, Page 152

Chapter 8

Multiple Moving Objects

In this chapter as well as the next three, we investigate various extensions of the basic motion planning problem. In the present chapter we begin this investigation with a series of extensions which all involve multiple moving objects.

We first consider the case where some (or all) of the obstacles in the workspace are moving (Section 1). The resulting problem is called the *dynamic motion planning problem.* Unlike the (static) basic problem, it cannot be solved by merely constructing a geometric path. Instead, a continuous function of time specifying the robot's configuration at each instant has to be generated. The general approach taken in this chapter is to add a dimension — time — to the robot's configuration space. The robot maps in this *configuration-time space* to a point moving among stationary obstacles. But the time dimension is irreversible and hence is significantly different from the other dimensions. Several path planning methods presented in the previous chapters are still applicable to the dynamic motion planning problem, but with some modifications and extensions, which are aimed at taking the specificity of the time dimension into account.

TESLA EXHIBIT 1013, Page 356

The dynamic motion planning problem can be made more realistic by imposing constraints on the robot's velocity and acceleration. In our presentation, however, we will only consider the two "simple" cases where (1) the robot's velocity modulus is not upper-bounded, and (2) where it is upper-bounded (with no other constraint on the velocity and the acceleration). As we will see, from a computational point of view, the problem with bounded velocity is substantially harder than the problem without bounded velocity.

As a second extension, we then consider the *multi-robot path planning problem* where several robots move independently in the same workspace among stationary obstacles (Section 2). One conceptually simple approach to this problem is to consider the various robots as the components of a composite robot and plan a free path in the configuration space of this composite robot (*centralized planning*). A difficulty, of course, is that the dimension of this configuration space is usually large and we know that the time complexity of path planning methods is exponential in this dimension. Another approach consists of planning a path for each robot more or less independently of the other robots and then considering the interactions among the paths (*decoupled planning*). By doing so, we significantly reduce the amount of computation, but we also loose completeness.

In the third extension, we consider *articulated robots*, i.e. robots which are made of several moving rigid objects connected by joints (e.g. hinges, sliding joints) that constrain the relative movements of the objects (Section 3). We focus our presentation on the case where the robots contain no kinematic loop. This case includes the problem of planning collision-free paths for manipulator arms.

1 Moving Obstacles

1.1 Configuration-Time Space

Let $\mathcal{A}$ be a rigid object moving in a workspace $\mathcal{W} = \mathbf{R}^N$, with $N = 2$ or 3, populated by obstacles $\mathcal{B}_i$ (rigid objects), $i = 1$ to q, which may possibly be in motion. $\mathcal{B}_i(t)$ designates the region of $\mathcal{W}$ occupied by the object $\mathcal{B}_i$ at time t ($t \geq 0$). We assume that the regions $\mathcal{B}_1(t)$ through $\mathcal{B}_q(t)$ are known in advance. This assumption requires that the motions

of the obstacles not be influenced by the motion of $\mathcal{A}$.

Let $\mathcal{C}$ be the configuration space of $\mathcal{A}$ as defined in Chapter 2. At every instant t, the obstacle $\mathcal{B}_i(t)$ maps in $\mathcal{C}$ to the C-obstacle:

$$\mathcal{CB}_i(t) = \{\mathbf{q} \in \mathcal{C} \;/\; \mathcal{A}(\mathbf{q}) \cap \mathcal{B}_i(t) \neq \emptyset\}.$$

The shape of $\mathcal{CB}_i(t)$ does not depend on t, but its location in $\mathcal{C}$ does.

The dynamic motion planning problem is to plan a collision-free motion of $\mathcal{A}$ from the initial configuration $\mathbf{q}_{init}$ at time 0 to the goal configuration $\mathbf{q}_{goal}$ at time $T \geq 0$. (We assume without loss of generality that the motion of $\mathcal{A}$ starts at time 0.) The time T is called the **arrival time.** The planning problem requires that a function of time be generated which specifies the configuration of $\mathcal{A}$ at every instant in the interval $[0, T]$. More formally, a solution of the problem is a continuous function:

$$t \in [0,T] \;\; \mapsto \mathbf{q}(t) \;\in\; \mathcal{C} \backslash \bigcup_{i=1}^{q} \mathcal{CB}_i(t)$$

with t representing time, $\mathbf{q}(0) = \mathbf{q}_{init}$ and $\mathbf{q}(T) = \mathbf{q}_{goal}$. This function is called a **trajectory** of $\mathcal{A}$.

Since the locations of the C-obstacles may vary with t, we cannot represent them in $\mathcal{C}$ in such a way that we can still reduce the motion of $\mathcal{A}$ to the motion of a point among fixed constraints. This difficulty is solved by simply adding another dimension — namely, time — to $\mathcal{C}$. We obtain a new space, $\mathcal{CT} = \mathcal{C} \times [0, +\infty)$, which is called the **configuration-time space**[1] of $\mathcal{A}$. $\mathcal{CT}$ is a smooth manifold of dimension $m + 1$, where m is the dimension of $\mathcal{C}$. A continuous map $\xi : [0,1] \rightarrow \mathcal{CT}$ is called a **CT-path.**

The robot $\mathcal{A}$ maps in $\mathcal{CT}$ to a **configuration-time** $(\mathbf{q}, t)$, meaning that $\mathcal{A}$'s configuration at instant t is $\mathbf{q}$. The cross-section through $\mathcal{CT}$ at every instant t is $\mathcal{C}$ populated by the C-obstacles $\mathcal{CB}_i(t)$, $i = 1$ to q. Thus, every obstacle $\mathcal{B}_i$ maps in $\mathcal{CT}$ to a stationary region $\mathcal{CTB}_i$, called a **CT-obstacle**, defined by:

$$\mathcal{CTB}_i = \{(\mathbf{q}, t) \;/\; \mathcal{A}(\mathbf{q}) \cap \mathcal{B}_i(t) \neq \emptyset\}.$$

[1] In the literature, this space is often called *configuration space-time.*

TESLA EXHIBIT 1013, Page 358

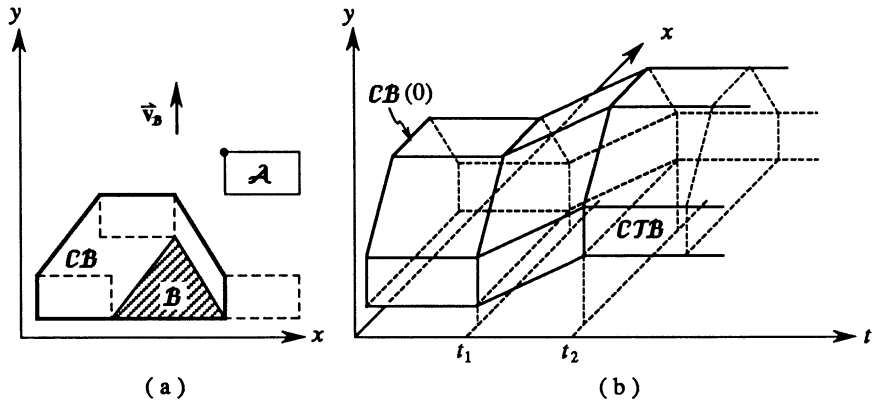


Figure 1. The robot $\mathcal{A}$ translates freely without rotation in the plane (Figure a). The obstacle $\mathcal{B}$ translates at fixed orientation, at constant linear velocity $\vec{\mathbf{v}}_{\mathcal{B}}$, between the instants t_1 and t_2. $\vec{\mathbf{v}}_{\mathcal{B}}$ is parallel to the y-axis and points upward. At any time $t \notin [t_1, t_2]$, $\mathcal{B}$ is motionless. The CT-obstacle $\mathcal{CTB}$ is obtained (see Figure b) by sweeping $\mathcal{CB}$ along a line that is perpendicular to the xy-plane, when $t \notin [t_1, t_2]$, and oriented along the vector $(0, \|\vec{\mathbf{v}}_{\mathcal{B}}\|, 1)$, when $t \in [t_1, t_2]$.

Example: Let $\mathcal{W} = \mathbf{R}^2$ and assume that $\mathcal{A}$ can only translate at fixed orientation. Hence, $\mathcal{C} = \mathbf{R}^2$ and $\mathcal{CT} = \mathbf{R}^2 \times [0, +\infty)$. Let the workspace contain a moving obstacle $\mathcal{B}$ that translates at fixed orientation, at constant linear velocity $\vec{\mathbf{v}}_{\mathcal{B}}$, between the instants t_1 and t_2. The rest of the time, $\mathcal{B}$ is stationary. Let $\mathcal{CB}(0)$ be the C-obstacle in $\mathcal{C}$ at instant 0. For all $t \in [t_1, t_2]$, $\mathcal{CB}(t)$ is obtained by translating $\mathcal{CB}(0)$ by $\vec{\mathbf{v}}_{\mathcal{B}}(t - t_1)$ in $\mathcal{C}$. Thus:

$$\begin{aligned} \mathcal{CTB} \;=\; & \{(\mathbf{q}, t) \,/\, \mathbf{q} \in \mathcal{CB}(0) \text{ and } t \in [0, t_1]\} \\ & \cup \{(\mathbf{q} + \vec{\mathbf{v}}_{\mathcal{B}}(t - t_1), t) \,/\, \mathbf{q} \in \mathcal{CB}(0) \text{ and } t \in [t_1, t_2]\} \\ & \cup \{(\mathbf{q} + \vec{\mathbf{v}}_{\mathcal{B}}(t_2 - t_1), t) \,/\, \mathbf{q} \in \mathcal{CB}(0) \text{ and } t \geq t_2\}. \end{aligned}$$

Figure 1 illustrates the construction of $\mathcal{CTB}$. ∎

We define the **free space** in $\mathcal{CT}$ as:

$$\mathcal{CT}_{free} = \mathcal{CT} \setminus \bigcup_{i=1}^{q} \mathcal{CTB}_i.$$

The closure of $\mathcal{CT}_{free}$ is called the **semi-free space** in $\mathcal{CT}$. A continuous map $\xi : [0, 1] \rightarrow \mathcal{CT}_{free}$ (resp. $cl(\mathcal{CT}_{free})$) is called a **free** (resp.

semi-free) CT-path. The dynamic motion planning problem can now be regarded as that of constructing a free CT-path between two given configuration-times $(\mathbf{q}_{init}, 0)$ and $(\mathbf{q}_{goal}, T)$. Hence, it is reduced to the problem of finding a path among stationary obstacles, i.e. the $\mathcal{CTB}_i$'s.

With respect to the basic motion planning problem, there is a major difference. Since no object can move back in time, a CT-path ξ should (at least) be *strictly monotone in time*, i.e. satisfy the following condition:

$$\forall s, s' \in [0,1], \text{ let } \xi(s) = (\mathbf{q}, t) \text{ and } \xi(s') = (\mathbf{q}', t');\ s < s' \Rightarrow t < t'.$$

Then ξ determines a trajectory $\mathbf{q}(t)$ such that: $\forall s \in [0,1] : \xi(s) = (\mathbf{q}(t), t)$.

In practice, the condition that a CT-path ξ be strictly time-monotone may not be sufficient, since it allows for unbounded velocity and acceleration. A variant of the dynamic motion planning problem consists of imposing constraints on $\mathcal{A}$'s velocity and acceleration, that is, on the slope and the curvature of ξ relative to the time axis. These constraints make the planning problem considerably harder. In the following, we will only consider the two cases where the robot's velocity modulus is/is not bounded. We will not examine the case where the acceleration modulus is bounded.[2]

Another classical variant of the dynamic motion planning problem is to leave the arrival time unspecified. This means that we only want $\mathcal{A}$ to reach a goal configuration $\mathbf{q}_{goal}$ at some time $T \geq 0$. Most of the methods solving the dynamic problem with specified arrival time T can solve the problem with unspecified T, with minor adaptations.

1.2 Complexity Results

It has been shown in [Reif and Sharir, 1985] that the dynamic motion planning problem in a three-dimensional workspace, with arbitrarily many rotating obstacles, is PSPACE-hard if the velocity modulus is bounded and NP-hard otherwise. (Notice that rotating obstacles may generate non-algebraic constraints in the configuration-time space, even if the objects are semi-algebraic.) It has also been shown in [Canny,

[2] Ó'Dúnlaing [Ó'Dúnlaing, 1987] is one of the rare publications considering the case where the robot's acceleration is bounded.

TESLA EXHIBIT 1013, Page 360

1988] that dynamic motion planning for a point in the plane, with a bounded velocity modulus and arbitrarily many obstacles, is NP-hard, even when the moving obstacles are convex polygons moving without rotation at constant linear velocities.

On the other hand, the dynamic motion planning problem for a polygonal robot $\mathcal{A}$ translating at fixed orientation in the plane with bounded velocity modulus among polygonal obstacles that translate without rotation at fixed linear velocity can be solved by an algorithm that is polynomial in the total number of vertices of $\mathcal{A}$ and the $\mathcal{B}_i$'s, if the number of obstacles is bounded (see Subsection 1.4.1). However, as one could suspect from the results cited above, the algorithm takes exponential time in the number of obstacles. When all the constraints can be expressed semi-algebraically, the dynamic motion planning problem with no velocity constraints and arbitrarily many obstacles can be solved in polynomial time in the number and the degree of the semi-algebraic constraints (see Subsection 1.3.1).

1.3 Planning Without Velocity Bound

1.3.1 Exact Cell Decomposition

When the modulus of $\mathcal{A}$'s velocity is not bounded and the obstacles' trajectories are expressed in algebraic form, the general path planning algorithm based on the Collins decomposition presented in Section 3 of Chapter 5 is applicable with minor changes. The various CT-obstacles are represented as semi-algebraic sets in $\mathcal{CT}$ and planning is performed with the additional constraint that time is irreversible.

This constraint can be taken into account by representing a configuration-time as a list $(t, q_1, ..., q_m) \in \mathbf{R}^{m+1}$ (m is the dimension of $\mathcal{A}$'s configuration space), where the time t stands as the first coordinate in the charts put on $\mathcal{CT}$. Hence, all the cells resulting from the Collins decomposition of $\mathcal{CT}$ ultimately project on the time axis. In other words, they are contained in "cylinders" above intervals or points along the time axis. Therefore, two cells κ and κ' are in the same cylinder, i.e. "occur" in the same interval of time, if and only if the sample points $\sigma(\kappa)$ and $\sigma(\kappa')$ project at the same time coordinate. If, instead, the sample points project at different time coordinates, one cell "occurs" before the other. The connectivity graph CG is now defined as a *directed*

graph whose nodes are the $(m+1)$- and m-dimensional cells of the Collins decomposition of $\mathcal{CT}$. A cell κ is connected to a cell κ' by an arc if and only if κ and κ' are adjacent and either κ occurs before κ', or κ and κ' occur simultaneously, i.e. the time coordinate of $\sigma(\kappa)$ is less than or equal to the time coordinate of $\sigma(\kappa')$.

In order to construct a free motion between $(\mathbf{q}_{init}, 0)$ and $(\mathbf{q}_{goal}, T)$, the planner searches CG for a sequence of adjacent cells $\kappa_1, ..., \kappa_p$ such that $(\mathbf{q}_{init}, 0) \in \kappa_1$ and $(\mathbf{q}_{goal}, T) \in \kappa_p$. From this sequence, if one exists, it generates a CT-path from $(\mathbf{q}_{init}, 0)$ to $(\mathbf{q}_{goal}, T)$ in the same way as for the static problem. Clearly, however, the slope of such a path with respect to the time axis is not bounded.

If the arrival time T is unspecified, the same method still applies. The goal locus in $\mathcal{CT}$ is the half-line $\{(\mathbf{q}_{goal}, T) \;/\; T \geq 0\}$ rather than a single configuration-time. The graph CG has to be searched for a sequence of cells $\kappa_1, ..., \kappa_p$ such that κ_p intersects this half-line. A search of CG with time as the cost function can generate the CT-path that attains the goal configuration at the earliest possible time, if a free CT-path exists.

This general method is complete, but its computational complexity makes it impractical, even for small values of m.

1.3.2 Approximate Cell Decomposition

The approximate cell decomposition approach presented in Chapter 6 also applies to the dynamic motion planning problem with no velocity constraint. It only requires the definition of the connectivity graph to be slightly modified.

Assume that the set of possible configuration-times in $\mathcal{CT}$ is a rectangloid which has been decomposed into smaller rectangloid cells, each classified as EMPTY (if its interior intersects no CT-obstacle), FULL (if it is entirely contained in the union of the CT-obstacles), or MIXED (otherwise). The connectivity graph CG corresponding to this decomposition is a directed graph whose nodes are the EMPTY cells (or both the EMPTY and MIXED cells) of the decomposition. A cell κ is connected to a cell κ' by an oriented arc if and only if:

- either they are adjacent by a face that is parallel to the time axis,

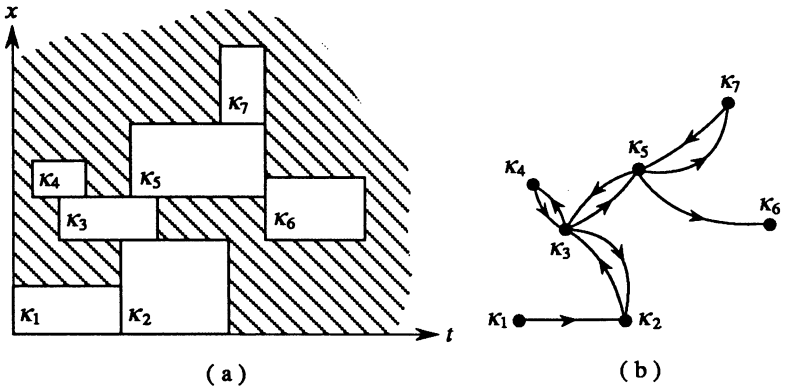


Figure 2. Figure a depicts a two-dimensional configuration-time space with seven EMPTY cells. Figure b shows the corresponding connectivity graph. Although no path from κ_2 to κ_4 can be time-monotone, this fact is not represented in the graph. Therefore, the search must be appropriately constrained.

- or they are adjacent by a face that is perpendicular to the time axis, and κ projects on the time axis before κ'.

Planning is accomplished by searching *CG* for a sequence of adjacent cells forming an EMPTY or MIXED channel such that the first cell of the sequence contains the initial configuration-time and the last cell contains the goal configuration-time.

Figure 2 illustrates this refinement with a two-dimensional configuration-time space. Seven EMPTY cells, κ_1 through κ_7, and the corresponding connectivity graph are depicted in the figure.

This refinement of the connectivity graph is not sufficient, however, to guarantee that the constructed channel contains a strictly time-monotone CT-path. It is necessary to put an additional constraint on the search of the graph. This need is illustrated in the graph of Figure 2. Although there exists a strictly time-monotone path between κ_2 and κ_3 and a strictly time-monotone path between κ_3 and κ_4, there is no such path between κ_2 and κ_4. This fact is not represented in the connectivity graph because it depends on which cells were traversed before κ_4 was entered, and hence involves more than two cells. The planner can deal with this difficulty during graph searching by maintaining the earliest time $t_{earliest}$ at which it can be in the current cell [Erdmann and

Lozano-Pérez, 1986]. Let $[t_{min}(\kappa), t_{max}(\kappa)]$ be the time interval on which the cell κ projects itself on the time axis. $t_{earliest}$ is simply computed as follows:

- when the planner starts the search and is in the initial cell:

$$t_{earliest} \leftarrow 0,$$

- when the planner decides to move from a cell κ to a cell κ':

$$t_{earliest} \leftarrow \max\{t_{earliest}, t_{min}(\kappa')\}.$$

Suppose that the planner has attained κ. It can choose κ' as the successor of κ in the search if and only if there is an arc from κ to κ' in CG and $t_{earliest} < t_{max}(\kappa')$. Consider Figure 2 and suppose that the search starts at κ_1. The earliest time in κ_1 is 0. The search can proceed to κ_2 with $t_{earliest} = t_{min}(\kappa_2)$, then to κ_3 with $t_{earliest}$ still equal to $t_{min}(\kappa_2)$. κ_3 has two successors, κ_4 and κ_5, in CG. However, κ_4 cannot be selected as a successor of κ_3 in the search, since $t_{earliest} > t_{max}(\kappa_4)$.

If an EMPTY channel is constructed, a free strictly time-monotone CT-path can easily be extracted from it.

For a configuration space of given dimension m, the time complexity of the approach is increased by the fact that it is now applied in the $(m+1)$-dimensional configuration-time space. But the constraint on the search does not increase the complexity further. As a matter of fact, it may only reduce the size of the search space.

1.4 Planning With Velocity Bound

In this subsection we consider the case where the modulus of the robot's velocity is upper-bounded. There is no constraint, however, on the robot's acceleration.

1.4.1 Asteroid Avoidance Problem

We first consider a rather specific type of dynamic motion planning problem called the two-dimensional *asteroid avoidance problem* [Reif and Sharir, 1985] and we describe a complete planning method for this problem. Roughly speaking, the method is a kind of visibility graph method extended in the robot's configuration-time space.

TESLA EXHIBIT 1013, Page 364

The robot is a polygon that can translate freely in $\mathbf{R}^2$ without rotation. Hence, $\mathcal{CT} = \mathbf{R}^2 \times [0, +\infty)$. The obstacles $\mathcal{B}_i$, $i = 1, ..., q$, are polygons that translate without rotation at fixed velocities (which may vary from one obstacle to another). The modulus $\|\vec{\mathbf{v}}_{\mathcal{A}}\|$ of the linear velocity of $\mathcal{A}$ is upper-bounded by V_{max}. In $\mathcal{A}$'s configuration space, the problem is equivalent to planning the motion of a point among polygonal C-obstacles that translate at constant linear velocities. We assume that the C-obstacles never intersect themselves. As mentioned in Subsection 1.2, this problem is NP-hard when the number of obstacles is not bounded.

Each CT-obstacle is obtained by sweeping the corresponding C-obstacle along a straight line whose slope relative to the time axis is determined by the constant linear velocity of the obstacle. The straight lines traced in $\mathcal{CT}$ by the C-obstacle's vertices are called the **vertex lines**.

We use the following terminology [Reif and Sharir, 1985]:

- A **direct motion** is a motion of $\mathcal{A}$ at constant velocity (possibly zero) such that $\mathcal{A}$ makes no contact with the obstacles, except possibly at the endpoints of the motion. In addition, at every endpoint where the motion makes a contact with an obstacle $\mathcal{B}_i$, the straight CT-path representing the direct motion in $\mathcal{CT}$ is supported by a line that is tangent at the contact point to the corresponding CT-obstacle $\mathcal{CTB}_i$.
- A **contact motion** is a motion of $\mathcal{A}$ along a convex subset (possibly a single point) of the boundary of a C-obstacle $\mathcal{CB}_i$, starting and ending at vertices of $\mathcal{CB}_i$. In $\mathcal{CT}$, the motion is thus described by a CT-path (possibly of length 0) which is contained in the boundary of $\mathcal{CTB}_i$ and begins and ends in a vertex line.
- A **fundamental motion** is a motion of $\mathcal{A}$ that consists of a direct motion possibly followed by a contact motion.
- A **normal motion** is a motion of $\mathcal{A}$ that consists of a sequence of fundamental motions such that any two direct motions are separated by a contact motion and no two contact motions are performed within the same maximal convex subset of the boundary of a C-obstacle. (Hence, if a C-obstacle is convex, a normal motion can visit it at most once.)

PROPOSITION 1: *There exists a time-monotone semi-free CT-path of $\mathcal{A}$ from $(\mathbf{q}_{init}, 0)$ to $(\mathbf{q}_{goal}, T)$ satisfying the velocity constraint*

$\|\vec{v}_A\| \leq V_{max}$ if and only if there exists a normal motion from $(q_{init}, 0)$ to (q_{goal}, T) satisfying the velocity constraint.

Proof: Obviously, a normal motion satisfying the velocity constraint produces a strictly time-monotone semi-free CT-path.

For the converse part, let the metric in $\mathcal{CT} \subset \mathbf{R}^3$ be the Euclidean metric in $\mathbf{R}^3$. If there exists a time-monotone semi-free CT-path ξ connecting two configuration-times that satisfies the velocity constraint, then the shortest CT-path ξ_{min} homotopic to ξ in $cl(\mathcal{CT}_{free})$ also satisfies the velocity constraint. We show below that ξ_{min} describes a normal motion.

The CT-path ξ_{min} must be such that any subpath ξ' of ξ_{min} is also the shortest path (in its homotopy class) between the two configuration-times that ξ' connects. Hence, at any configuration-time in $cl(\mathcal{CT}_{free})$ that is not in a vertex line, ξ_{min} must have zero curvature, since otherwise there is locally a shorter subpath. At any configuration-time contained in a vertex line, ξ_{min} must locally (i.e. in a sufficiently small neighborhood) be either a straight line or a polygonal line with a single vertex (the considered configuration-time); if it is a polygonal line, then the CT-obstacle must lie in its convex sector. (In particular, this implies that ξ_{min} does not traverse any vertex line generated by a C-obstacle's concave vertex.) Finally, consider a subpath ξ' of ξ_{min} whose endpoints lie in the same maximal convex subset S of the boundary of a CT-obstacle; ξ' must lie in S, be contained in a plane, and thus consist of successive straight segments. Hence, ξ_{min} cannot visit S twice.

Therefore, ξ_{min} describes a normal motion. ∎

The shortest semi-free CT-path in some homotopy class between two free configuration-times may lie partly in the boundary of $\mathcal{CT}_{free}$. Since the CT-obstacles are disjoint, one can obtain a free CT-path by an arbitrarily small perturbation of the shortest semi-free CT-path. But such a free CT-path may violate the velocity constraint by an arbitrarily small offset. Indeed, assume that the straight segment connecting $(q_{init}, 0)$ to (q_{goal}, T) lies in $cl(\mathcal{CT}_{free})$ and that its slope relative to the time axis is exactly V_{max}. Assume further that it makes a single contact with a CT-obstacle. Then, in order to avoid the contact, the robot would have to go faster, either before or after it passes near the obstacle. An alternative to violating the constraint is to slightly shift either the initial or

TESLA EXHIBIT 1013, Page 366

the goal configuration-time.

From the above proposition, we can infer the planning method presented below, which is due to Reif and Sharir [Reif and Sharir, 1985]. We first outline the algorithm. Then we analyze some of its components in more detail. For simplifying the presentation, we assume that all the C-obstacles are convex polygons. However, it is not difficult to extend the algorithm to non-convex polygonal C-obstacles.

Let q be the number of obstacles in the environment, Q_i the set of vertices of $\mathcal{CB}_i$, $Q_0 = \{\mathbf{q}_{init}\}$, $Q_{q+1} = \{\mathbf{q}_{goal}\}$, and $Q = \bigcup_{i=0}^{q+1} Q_i$. For any point b in Q, we denote by $b(t)$ its position in $\mathcal{C}$ at time t, by $\overline{b}$ the (half-)line $\{b(t) \,/\, t \in [0, +\infty)\}$ to which b maps in $\mathcal{CT}$, and by $\overline{Q}$ the set $\{\overline{b} \,/\, b \in Q\}$. Since $\mathbf{q}_{init}$ and $\mathbf{q}_{goal}$ are stationary points in $\mathcal{C}$, $\overline{\mathbf{q}}_{init}$ and $\overline{\mathbf{q}}_{goal}$ are lines parallel to the time axis of $\mathcal{CT}$. In the following, we only consider motions that satisfy the velocity constraint.

Let $I_b^{(k)}$ denote the set of all times[3] $t \geq 0$ at which the line $\overline{b} \in \overline{Q}$ can be attained from the initial configuration-time $(\mathbf{q}_{init}, 0)$ by a motion consisting of at most k fundamental motions. We have:

$$I_{\mathbf{q}_{init}}^{(0)} = [0, +\infty) \quad \text{and} \quad I_b^{(0)} = \emptyset$$

for every $b \in Q \backslash \{\mathbf{q}_{init}\}$.

Let $DM_{b,b'}(I)$ be the set of all times $t' \geq 0$ such that $\overline{b'} \in \overline{Q}$ can be reached at time t' by a single direct motion starting at the configuration-time $(b(t), t)$, with $t \in I$. For every $b' \in Q$,

$$J_{b'}^{(k)} = \bigcup_{b \in Q} DM_{b,b'}(I_b^{(k)})$$

is the set of all times at which $\overline{b'}$ can be reached from $(\mathbf{q}_{init}, 0)$ by a motion consisting of at most k fundamental motions followed by a direct motion.

Let $CM_{b',b''}(I)$, with b' and b'' being in the same set Q_j for some $j \in [1, q]$, be the set of all times $t'' \geq 0$ for which $\overline{b''}$ can be attained at time t'' by

[3] It would be slightly more efficient to restrict t to the interval $[0, T]$ rather than $[0, +\infty)$. However, by making the presentation more general, the same algorithm applies to the case where the arrival time T is unspecified.

a contact motion in the boundary of $\mathcal{CTB}_j$ starting at $(b'(t'), t')$, with $t' \in I$. For every $b'' \in Q_j$ ($j \in [1, q]$), we have:

$$I_{b''}^{(k+1)} = \bigcup_{b' \in Q_j} CM_{b',b''}(J_{b'}^{(k)}).$$

Assume that we know how to compute the sets $DM_{b,b'}(I)$ and $CM_{b',b''}(I)$. Then, for $k = 1, 2, ...$, we can successively compute the sets $I_b^{(k)}$ for all $b \in Q \backslash \{\mathbf{q}_{init}\}$, until either $T \in I_{\mathbf{q}_{goal}}^{(k)}$, or $k > q + 1$. In the first case, the problem is solved and a semi-free CT-path can be generated by moving backward along the time axis from $(\mathbf{q}_{goal}, T)$. In the second case, the problem admits no solution; indeed, according to the above proposition, if a semi-free CT-path exists, there exists a normal motion that visits each CT-obstacle at most once, and hence consists of at most $q + 1$ fundamental motions. If the arrival time T is not specified, the same algorithm applies; it terminates when either $I_{\mathbf{q}_{goal}}^{(k)} \neq \emptyset$ (success) or $k > q + 1$ (failure).

The computation of $CM_{b,b'}(I)$ is simple and not described here. The number of disjoint intervals in $CM_{b,b'}(I)$ is less than or equal to the number of disjoint intervals in I.

The computation of $DM_{b,b'}(I)$ is more involved. Consider $DM_{b,b'}(\{t\})$, i.e. the set of all times t' such that $\overline{b'}$ can be reached by a single direct motion starting at $(b(t), t)$. The set of CT-paths representing the direct motions starting at $(b(t), t)$ spans a one-sided cone of angle $\tan^{-1}(V_{max})$. This cone is anchored at $(b(t), t)$ and its axis points along the time axis. Its intersection with the line $\overline{b'}$ is a segment (possibly semi-infinite or empty) joining two points $(b'(t_1), t_1)$ and $(b'(t_2), t_2)$, with t_1 and t_2 depending on t. We denote by $L(t)$ the interval $[t_1, t_2]$. We have that $DM_{b,b'}(\{t\}) \subseteq L(t)$, since for some $t' \in L(t)$ the straight CT-path between $(b(t), t)$ and $(b(t'), t')$ may traverse the interior of a CT-obstacle. Let us consider the triangle Δ in $\mathcal{CT}$, whose vertices are $(b(t), t)$, $(b(t_1), t_1)$ and $(b(t_2), t_2)$. Each CT-obstacle $\mathcal{CTB}_i$ ($i \in [1, q]$) intersects Δ at a (possibly empty) convex polygon. The two rays drawn from $(b(t), t)$ which are tangent to this polygon cut a segment $S_i(t)$ off $L(t)$. $S_i(t)$ is the set of configuration-times in $\overline{b'}$ that are not reachable from $(b(t), t)$ by a direct motion due to $\mathcal{CTB}_i$. Let $U_i(t)$ be the projection of $S_i(t)$ on the time axis of $\mathcal{CT}$. Let $F_{b,b'}$ denote the set of all pairs of

TESLA EXHIBIT 1013, Page 368

times (t, t') such that $(b'(t'), t')$ can be reached from $(b(t), t)$ by a single direct motion. From the above discussion, we have:

$$F_{b,b'} = \{(t, t') \in [0, +\infty) \times [0, +\infty) \ / \ t' \in L(t) \setminus \bigcup_{i=1}^{q} U_i(t)\}.$$

It is a two-dimensional region and its intersection by a line at constant t consists of $q+1$ intervals at most. Suppose it has been computed. Then:

$$DM_{b,b'}(I) = \{t' \in [0, +\infty) \ / \ \exists t \in I \text{ such that } (t, t') \in F_{b,b'}\}.$$

We now sketch the computation of $F_{b,b'}$. In the space $\{(t, t') \in [0, +\infty) \times [0, +\infty)\}$, the endpoints of the interval $L(t)$ trace an algebraic curve of maximal degree 2, whose equation is obtained by intersecting the fixed line $\overline{b'}$ with the surface of the cone spanned by the CT-paths originating at $b(t)$. This curve bounds a domain D which includes $F_{b,b'}$. Now for each C-obstacle vertex b'', we compute the time coordinate $e_{b''}(t)$ of the intersection[4] of the plane containing both $(b(t), t)$ and $\overline{b''}$ with the line $\overline{b'}$. The curve $t' = e_{b''}(t)$ is an algebraic curve of maximal degree 2. We can compute the intersections of the various curves $t' = e_{b''}(t)$ inside D, determine the regions bounded by these curves and the boundary of D, and label each of these regions according to whether it is contained in $F_{b,b'}$ or not. (A sweep-line technique can perform these computations and that of $DM_{b,b'}(I)$ concurrently.)

Let n be the total number of vertices in the C-obstacles. There are $O(n^2)$ intersections among the curves $t' = e_{b''}(t)$, and between these curves and the boundary of D. Therefore, there are at most $O(n^2)$ regions in D, and $F_{b,b'}$ can be computed in polynomial time. It follows that the computation of every set $I_b^{(k)}$ also takes time polynomial in n, but exponential in k (since $I_b^{(k)}$ contains $O(q^k)$ disjoint intervals in the worst case). Hence, in the worst case where $I_{\mathbf{q}_{goal}}^{(q+1)}$ has to be computed, the overall algorithm requires time polynomial in n and exponential in q. If q is fixed, the algorithm is polynomial.

1.4.2 Approximate Cell Decomposition

In Subsection 1.3.2 we described a modification of the approximate cell decomposition approach that solves the dynamic motion planning prob-

[4] Pathological cases can be avoided by imposing that all obstacles move at different non-zero velocities.

lem with no velocity constraint. This approach can be adapted further in order to deal with bounded velocity.

The principle of the adaptation is the following[5]. Concurrent with the generation of an EMPTY (or MIXED) channel, the planner constructs a time-monotone CT-path contained in this channel that satisfies the velocity constraint. One way to construct this path is to discretize each rectangloid cell in the channel into a grid of configuration-times and search the composite grid contained in the channel generated so far. Hence, the planner concurrently searches (1) the cell connectivity graph for a channel, and (2) the grid contained in the channel generated so far for a CT-path satisfying the velocity constraint. Whenever the planner considers the inclusion of a new cell at the end of the current channel, it verifies that a path can be extended to this cell without violating the velocity constraint. If this is not the case, the cell is not included in the channel.

Due to the discretization in each cell, the planning method is not guaranteed to find a CT-path satisfying the velocity constraint in a channel, even if one exists. Hence, it may lead to reject a solution channel and fail to solve the planning problem at hand, although a solution exists. The method can be made more reliable by using a finer discretization of the cells, but this also increases the computational cost of the method.

1.4.3 Velocity Tuning

Velocity tuning is a two-phase approach to the dynamic motion planning problem. This approach, which was introduced in [Kant and Zucker, 1986a], consists of first planning a path for $\mathcal{A}$ from $\mathbf{q}_{init}$ to $\mathbf{q}_{goal}$ among the stationary obstacles lying in $\mathcal{W}$, and then tuning the velocity of $\mathcal{A}$ along the path in order to avoid collisions with the moving obstacles. Hence, the original problem is broken down into two subproblems. The first subproblem is a basic motion planning problem that can be solved using a method chosen among those presented in Chapters 4 through 7. The second subproblem is a *velocity planning* problem. As we will see below, it can be formulated as a dynamic motion planning problem in a path-time space similar to a two-dimensional configuration-time space.

[5]This adaptation is derived from [Fujimura and Samet, 1989], where an implemented planner is described in detail.

TESLA EXHIBIT 1013, Page 370

Let $\tau : l \in [0, L] \mapsto \tau(l) \in \mathcal{C}'_{free}$ be the path planned in the first phase. $\mathcal{C}'_{free}$ denotes the set of configurations where $\mathcal{A}$ has no intersection with the stationary obstacles. The parameter l measures the length of the path between $\tau(0)$ and $\tau(l)$ (in the metric of $\mathcal{C}$). Hence, L denotes the total length of the path. The velocity planning problem is to generate a continuous function $\lambda : t \in [0, T] \mapsto \lambda(t) \in [0, L]$, with $\lambda(0) = 0$ and $\lambda(T) = L$, such that the composed map $\tau \circ \lambda : t \in [0, T] \mapsto \tau(\lambda(t))$ verifies $(\tau(\lambda(t)), t) \in \mathcal{CT}_{free}$ for all $t \in [0, T]$. The map λ must also be such that $\mathcal{A}$'s velocity be bounded by V_{max}. It may not be bijective, so that $\mathcal{A}$ may traverse some configurations in τ more than once.

Let the lt-space $[0, L] \times [0, T]$ be denoted by $\mathcal{PT}$. $\mathcal{PT}$ is called the **path-time space**. For a given path τ of $\mathcal{A}$ in $\mathcal{C}$, a moving obstacle $\mathcal{B}(t)$ maps in this space to a region $\mathcal{PTB}$, called a **PT-obstacle**, defined as follows:

$$\mathcal{PTB} = \{(l, t) \in [0, L] \times [0, T] \,/\, \tau(l) \in \mathcal{CB}(t)\}.$$

Let:

$$\mathcal{PT}_{free} = [0, L] \times [0, T] \setminus \bigcup_i \mathcal{PTB}_i$$

where the $\mathcal{PTB}_i$'s are the PT-obstacles corresponding to the moving obstacles in $\mathcal{W}$. The velocity planning problem is thus reduced to finding a strictly time-monotone continuous path in $\mathcal{PT}_{free}$, called a **free PT-path**, between $(0, 0)$ and (L, T). Indeed, this free PT-path determines a unique function $\lambda(t)$ such that $\forall t \in [0, T]$, $(\tau(\lambda(t)), t) \in \mathcal{CT}_{free}$. The slope of the PT-path with respect to the time axis must also satisfy the velocity constraint, i.e. $|\frac{dl}{dt}| \leq V_{max}$. (Hence, the velocity tuning approach corresponds to constraining the solution CT-path to lie in a two-dimensional surface swept out in $\mathcal{CT}$ by a line parallel to the time axis and following the path τ generated in $\mathcal{C}$. $\mathcal{PT}$ is the space obtained by "flattening" this surface on a plane.)

The PT-obstacles may have complicated shapes. A connected C-obstacle $\mathcal{CB}$ can even map in $\mathcal{PT}$ to a region made of several connected components. In the following, we will assume that the PT-obstacles are polygons. This representation is exact when the path τ is a sequence of straight segments in $\mathcal{C} = \mathbf{R}^2$ or $\mathbf{R}^3$ and the moving C-obstacles are polygons/polyhedra translating at constant linear velocities without rotation (asteroid avoidance problem). In other cases, it is obtained by approximating the PT-obstacles.

TESLA EXHIBIT 1013, Page 371

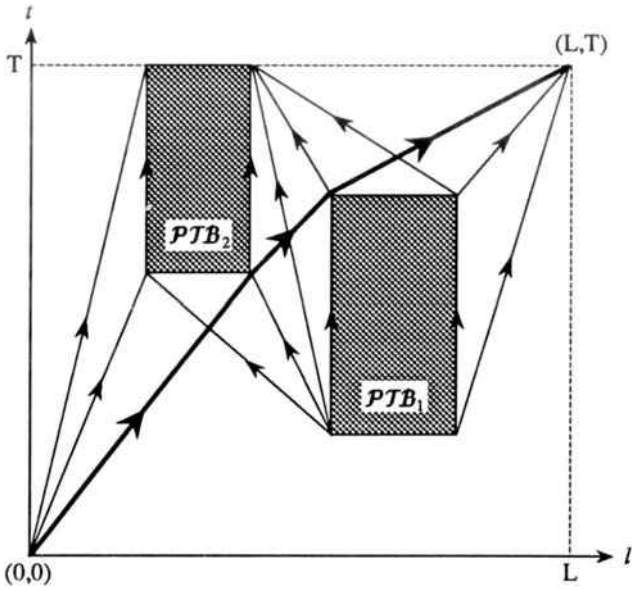


Figure 3. This figure displays a simple polygonal path-time space with two rectangular PT-obstacles and the corresponding search graph. This graph is a refinement of the reduced visibility graph that incorporates the irreversible nature of time and the upper-bound constraint on the robot's velocity modulus. The path connecting $(0,0)$ to (L,T) is shown in bold line.

With polygonal PT-obstacles, the velocity planning problem can easily be solved by using a variant of the visibility graph method presented in Section 1 of Chapter 4. The variant consists of searching a refinement of the reduced visibility graph. The new graph — let us denote it by G — is an oriented graph defined as follows. The nodes of G are the convex vertices of the PT-obstacles and the two points $(0,0)$ and (L,T). An arc of G connects the node (l,t) to (l',t') if and only if:

- $t < t'$, $|\frac{l'-l}{t'-t}| \leq V_{max}$, and
- either the line segment joining the two points is a PT-obstacle's edge, or it lies entirely in $\mathcal{PT}_{free}$, except possibly at each of the two endpoints, where it is tangent to a PT-obstacle.

One can show that there exists a semi-free PT-path connecting $(0,0)$ to (L,T) in $cl(\mathcal{PT}_{free})$ if and only if there exists a path in G between the

two corresponding nodes (see proof of Proposition 1 in Subsection 1.4.1). If a path in G entails a contact with some obstacles, an arbitrarily small perturbation makes it into a free PT-path (which may slightly violate the velocity constraint). Figure 3 shows a simple path-time space and the corresponding graph G. If $(0,0)$ and (L,T) are in the same connected component of G, the search of G allows us to construct a solution PT-path (shown in a bold line in Figure 3). Otherwise, the search terminates with failure.

The above method can be easily extended to the case where the arrival time T is unspecified.

The velocity tuning approach is inherently incomplete. Given a path of $\mathcal{A}$ that avoids collisions with the stationary obstacles, there may exist no velocity profile along that path that allows $\mathcal{A}$ to avoid collisions with the moving obstacles. This happens, for example, when an obstacle comes at the robot on approximately the same geometric path (for instance, in a narrow corridor). Perhaps another path might have made it possible to avoid collision with both the stationary and the moving obstacles. Whenever the velocity planning phase fails, the planner may try to generate another path among the stationary obstacles. But the approach does not provide a systematic way of choosing this other path.

The velocity tuning approach, which can nevertheless solve many practical problems, is substantially more time-efficient than systematic methods. It is polynomial (rather than exponential) in the number of C-obstacles and the total number of their vertices. In addition, it substitutes a twofold search in an m-dimensional configuration space and in a two-dimensional path-time space for a search in an $(m+1)$-dimensional configuration-time space.

2 Multiple Robots

In this section we consider a collection of p robots, $\mathcal{A}_i$, $i = 1, ..., p$, moving in the same workspace $\mathcal{W} = \mathbf{R}^N$, $N = 2$ or 3, among stationary obstacles $\mathcal{B}_j$, $j = 1, ..., q$. The robots move independently of each other, but they cannot occupy the same space at the same time. The problem is to plan a collision-free motion of the robots from an initial configuration to a goal configuration.

2.1 Composite Configuration Space

Let $\mathcal{C}_1$ through $\mathcal{C}_p$ denote the individual configuration spaces of the p robots $\mathcal{A}_1$ through $\mathcal{A}_p$. Each space $\mathcal{C}_i$, $i \in [1,p]$, contains two sorts of C-obstacles:

- C-obstacles corresponding to the collision of $\mathcal{A}_i$ with the stationary obstacles $\mathcal{B}_1$ through $\mathcal{B}_q$,
- C-obstacles corresponding to the collision of $\mathcal{A}_i$ with the other robots.

C-obstacles of the second sort dependent on the motion of the other robots and cannot be represented as fixed regions in $\mathcal{C}_i$.

The most direct way to extend the notion of configuration space to multiple robots is to consider the objects $\mathcal{A}_1$ through $\mathcal{A}_p$ as the components of the same composite robot $\mathcal{A} = \{\mathcal{A}_1, ..., \mathcal{A}_p\}$. Let $\mathcal{F}_{\mathcal{A}_i}$ be a Cartesian frame attached to $\mathcal{A}_i$. A **configuration q** of $\mathcal{A}$ is a specification of the position and orientation of every frame $\mathcal{F}_{\mathcal{A}_i}$ ($i = 1$ to p) with respect to the workspace frame $\mathcal{F}_{\mathcal{W}}$. Since the robots move independently of each other, a configuration of $\mathcal{A}$ is of the form $\mathbf{q} = (\mathbf{q}_1, ..., \mathbf{q}_p)$, where $\mathbf{q}_i \in \mathcal{C}_i$ denotes a configuration of $\mathcal{A}_i$, and we have: $\mathcal{A}(\mathbf{q}) = \mathcal{A}_1(\mathbf{q}_1) \cup ... \cup \mathcal{A}_p(\mathbf{q}_p)$. $\mathcal{A}$'s configuration space, $\mathcal{C} = \mathcal{C}_1 \times ... \times \mathcal{C}_p$, is called the **composite configuration space** of $\mathcal{A}_1$ through $\mathcal{A}_p$. It is a smooth manifold of dimension $m = \sum_{i=1}^{p} m_i$, where m_i denotes the dimension of $\mathcal{C}_i$.

There are still two sorts of C-obstacles in $\mathcal{C}$, but they are now both time independent:

- The C-obstacle due to the interaction of $\mathcal{A}_i$ with the obstacle $\mathcal{B}_j$ is denoted by $\mathcal{CB}_{ij}$ and defined by:

$$\mathcal{CB}_{ij} = \{\mathbf{q} = (\mathbf{q}_1, ..., \mathbf{q}_i, ..., \mathbf{q}_p) \in \mathcal{C} \;/\; \mathcal{A}_i(\mathbf{q}_i) \cap \mathcal{B}_j \neq \emptyset\}.$$

- The C-obstacle due to the interaction of $\mathcal{A}_i$ and $\mathcal{A}_j$, $i < j$, is denoted by $\mathcal{CA}_{ij}$ and defined by:

$$\mathcal{CA}_{ij} = \{\mathbf{q} = (\mathbf{q}_1, ..., \mathbf{q}_i, ..., \mathbf{q}_j, ..., \mathbf{q}_p) \in \mathcal{C} \;/\; \mathcal{A}(\mathbf{q}_i) \cap \mathcal{A}(\mathbf{q}_j) \neq \emptyset\}.$$

The computation of the $\mathcal{CA}_{ij}$'s can be done by a slight adaptation of the methods presented in Chapter 3.

The **free space** in $\mathcal{C}$ is defined by:

$$\mathcal{C}_{free} = \mathcal{C} \setminus \left(\bigcup_{i\in[1,p],j\in[1,q]} \mathcal{CB}_{ij} \right) \cup \left(\bigcup_{i\in[1,p],j\in[i+1,p]} \mathcal{CA}_{ij} \right).$$

A **free path** between two configurations $\mathbf{q}_{init}$ and $\mathbf{q}_{goal}$ is a continuous map $\tau : [0,1] \rightarrow \mathcal{C}_{free}$, with $\tau(0) = \mathbf{q}_{init}$ and $\tau(1) = \mathbf{q}_{goal}$.

Hence, the problem of planning a free path for multiple robots moving in the same workspace is formulated as the problem of planning a free path τ for a single composite robot. The projection of τ on every space $\mathcal{C}_i$, $i \in [1,p]$, is the corresponding path of $\mathcal{A}_i$. Although time does not appear explicitly, the execution of τ requires that the individual robots follow their own paths in a coordinated fashion. Changing the velocity of one robot requires the velocities of the other robots to be changed in the same ratio.

2.2 Centralized Planning

We call *centralized planning* the approach which consists of planning the coordinated paths of multiple robots as a path in their composite configuration space. This approach allows us to use general path planning methods described in Chapters 4 through 7.

As an illustration, Figure 4 shows several configurations along a path for two rectangular robots in a workspace with three narrow corridors. The two robots cannot pass each other in the same corridor. The planning problem is essentially that of interchanging the two robots in the central corridor. The path was generated using the randomized potential-field-based method (RPP) described in Section 5 of Chapter 7 in the six-dimensional composite configuration space of the two robots.

Several specific methods based on the approaches described in Chapters 4 through 7 have also been developed to solve particular multi-robot motion planning problems. For instance, Schwartz and Sharir [Schwartz and Sharir, 1983c] presented an exact cell decomposition method for planning the motion of two discs in the plane among polygonal obstacles. The method is based on the definition of critical curves (in the individual two-dimensional configuration spaces of the discs) where the possible contacts among the discs and the obstacles change qualitatively.

TESLA EXHIBIT 1013, Page 375

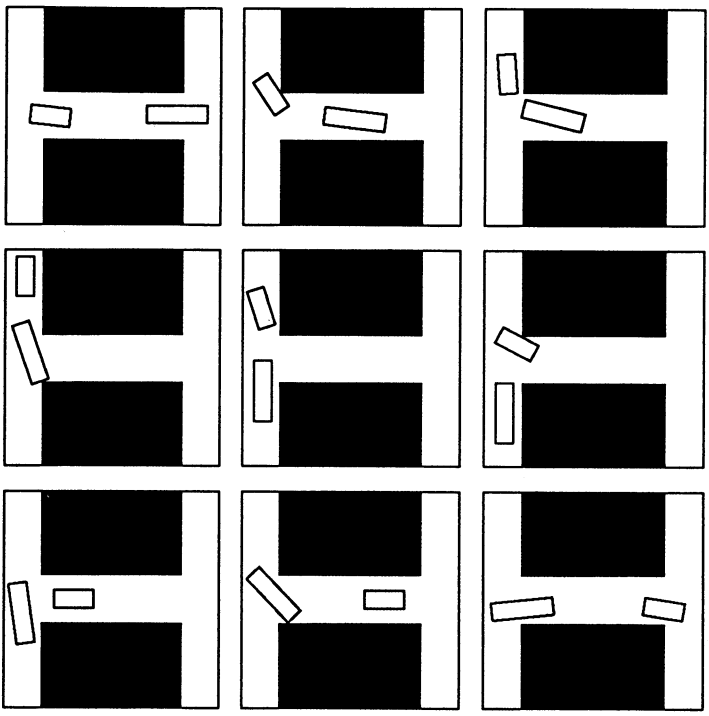


Figure 4. The planning problem is to interchange two robots in a workspace made of narrow corridors. The two robots cannot pass each other in the same corridor. The figure shows several snapshots along a path planned using a centralized planning approach with the randomized potential-field-based method described in Section 5 of Chapter 7. Both robots have to move away to intermediate configurations before actually moving to their goal configurations.

The method runs in time $O(n^3)$, where n is the number of obstacle edges. The same method can be extended to more than two discs, but its complexity rapidly increases with the number of discs. An $O(n^2)$ exact cell decomposition algorithm for two discs was also presented in [Sharir and Sifrony, 1988].

Barraquand, Langlois and Latombe [Barraquand, Langlois and Latombe, 1989b] described a potential field method for planning the motion of many discs in an environment including narrow corridors in which two discs cannot pass each other. The method concurrently builds and searches a local-minima graph of the potential function defined over

$\mathcal{C} = \mathbf{R}^{2p}$, where p is the number of discs. The specificity of the method rests on the technique used to escape local minima. At each encountered local minimum the method simulates a series of *constrained motions.* Each constrained motion is generated by forcing one configuration coordinate to increase or decrease until a saddle point of the potential function is attained. During the constrained motion, the other $m - 1$ coordinates are selected along the negative slope of the gradient of the potential. As an illustration, consider the case where a gradient motion leads two discs to move in opposite directions toward each other, in the same narrow corridor. At some point, the two discs cannot progress further toward their respective goals without colliding with each other. Then, a local minimum of the potential function has been attained by the gradient motion. A constrained motion corresponds to one of the discs moving backward, with the other disc still proceeding forward along the negated gradient until there is sufficient free room for the two discs to pass each other (saddle point of the potential). This planning method interweaving gradient and constrained motions was successfully experimented with up to 10 discs.

Tournassoud [Tournassoud, 1986] proposed a variant of the potential field approach where motion coordination is expressed as a local optimization problem. The basic idea is to force the moving objects to slide along tangent separating hyperplanes.

The practical difficulty with centralized planning is that it yields methods whose time complexity is exponential in the dimension of the composite configuration space, i.e. $m = \sum_{i=1}^{p} m_i$. Indeed, as mentioned in Section 6 of Chapter 1, path planning in an m-dimensional configuration space is PSPACE-hard when m is not bounded. Even the apparently simpler problem of coordinating the motion of arbitrarily many rectangular objects in an empty rectangular workspace is PSPACE-complete.

2.3 Decoupled Planning

One way to reduce the computational complexity of the multi-robot path planning problem is to first plan a path for each robot more or less independently of the other robots and then consider the interactions among the paths. We call such an approach *decoupled planning.* Roughly speaking, while centralized planning is exponential in $m = \sum_{i=1}^{p} m_i$, decoupled planning is exponential in $\max_{i \in [1,p]}\{m_i\}$. However, this gain

results in a loss of completeness. A decoupled planning approach may fail to plan a path, while a centralized planning approach using the same sort of path planning method would have succeeded (perhaps at the expense of a long computation).

Below, we describe two decoupled planning approaches, which we call *prioritized planning* and *path coordination.*

2.3.1 Prioritized Planning

Prioritized planning consists of considering the motions of the $\mathcal{A}_i$'s, one robot at a time, say from $i = 1$ to $i = p$ [Erdmann and Lozano-Pérez, 1986]. At each iteration, a path of $\mathcal{A}_i$ is generated in the configuration-time space $\mathcal{CT}_i$ of $\mathcal{A}_i$, avoiding collisions with both the obstacles $\mathcal{B}_j$, $j = 1, ..., q$, and the robots $\mathcal{A}_1, ..., \mathcal{A}_{i-1}$. Hence, the motion of $\mathcal{A}_i$ is planned as if the robot was moving among stationary obstacles and $i - 1$ moving obstacles. The path of $\mathcal{A}_1$ is planned in $\mathcal{C}_1$, rather than in $\mathcal{CT}_1$, and an arbitrary velocity is selected in order to map $\mathcal{A}_1$ into the configuration-time spaces of the other robots. This velocity merely determines a scale factor along the time axes of these spaces. Multiplying the velocity of $\mathcal{A}_1$ by a constant factor results in multiplying the other robots' velocities by the same factor. The path of $\mathcal{A}_1$ can be generated using an available path planning method. The motion of every $\mathcal{A}_i$, $i > 1$, can be generated using a planning method described in Section 1.

The ordering on the robots requires a priority to be assigned to each robot. Priorities may be assigned randomly. However, a better way is to infer priorities from motion constraints. For instance, a simple rule could be to plan the motions of the bigger robots before those of the smaller ones. Buckley [Buckley, 1989a] proposed to assign priorities by attempting to maximize the number of robots which can move in straight line from their initial configuration to their goal configuration (in Buckley's application the robots are translating, two-dimensional objects).

Note that the prioritized planning approach would have great difficulty in solving the planning problem of Figure 4. Indeed, the path of the first robot (say, the longest one) is likely to be the most direct one, i.e. along this path, the robot stays in the central corridor. This makes it impossible to find a path for the second robot, since the corridor is too narrow for the two robots to pass each other. The planner may try to

TESLA EXHIBIT 1013, Page 378

generate another path for the first robot, but the prioritizing planning approach offers no systematic way of choosing it. In addition, by allowing backtracking, the time complexity of the prioritized planning approach is substantially augmented. Thus, when the approach fails, rather than backtracking, it might be more suitable to call another algorithm based on the centralized planning approach, or to give up.

2.3.2 Path Coordination

Path coordination is a decoupled planning approach proposed in [O'Donnell and Lozano-Pérez, 1989]. It is based on a scheduling technique for dealing with limited resources which was originally developed for concurrent access to a database by several users [Yannakakis, Papadimitriou and Kung, 1979]. Here, space is the shared resource. The approach is applicable when the planning problem involves only two robots $\mathcal{A}_1$ and $\mathcal{A}_2$. It consists of first generating a free path for each of the two robots, independently of the other robot, and then coordinating the two paths so that the robots do not collide with each other:

Let:

$$\tau_1 : s_1 \in [0,1] \mapsto \tau_1(s_1) \in \mathcal{C}_{1,free}$$

$$\tau_2 : s_2 \in [0,1] \mapsto \tau_2(s_2) \in \mathcal{C}_{2,free}$$

be the two free paths generated for $\mathcal{A}_1$ and $\mathcal{A}_2$. $\mathcal{C}_{1,free}$ (resp. $\mathcal{C}_{2,free}$) denotes the free space in the configuration space of $\mathcal{A}_1$ (resp. $\mathcal{A}_2$) obtained by ignoring the other robot. Consider the $s_1 s_2$-space $[0,1] \times [0,1]$. Any continuous path in this space connecting $(0,0)$ to $(1,1)$ is called a **schedule** and determines a coordinated execution of the two paths. For instance, the schedule that first connects $(0,0)$ to $(1,0)$ by a straight segment and then $(1,0)$ to $(1,1)$ by another straight segment corresponds to the case where $\mathcal{A}_1$ moves first and completely executes the path τ_1 before $\mathcal{A}_2$ starts moving along τ_2. If s_1 (resp. s_2) is proportional to the length of the subpath from $\tau_1(0)$ to $\tau_1(s_1)$ (resp. from $\tau_2(0)$ to $\tau_2(s_2)$), then the schedule that connects $(0,0)$ to $(1,1)$ by a single straight segment — a diagonal of the rectangle $[0,1] \times [0,1]$ — corresponds to the case where the two robots move simultaneously, with a constant ratio of their velocity moduli.

Let us define the obstacle region in the $s_1 s_2$-space as the set of pairs (s_1, s_2) such that $\mathcal{A}(\tau_1(s_1)) \cap \mathcal{A}_2(\tau_2(s_2)) \neq \emptyset$. The second phase of the

path coordination approach is to construct a **free schedule**, i.e. a schedule that does not traverse this obstacle region. Hence, the path coordination approach corresponds to constraining the path of $\mathcal{A} = \{\mathcal{A}_1, \mathcal{A}_2\}$ in the composite configuration space $\mathcal{C} = \mathcal{C}_1 \times \mathcal{C}_2$ to lie in the intersection of two subspaces whose projections into $\mathcal{C}_1$ and $\mathcal{C}_2$ are τ_1 and τ_2, respectively. The $s_1 s_2$-space is obtained by "flattening" this intersection, which has dimension 2, on a plane.

Notice that the path coordination approach puts an even stronger constraint on the search of a solution than the prioritizing planning approach. Therefore, it may fail more often. When the path coordination approach fails, it seems appropriate to execute an algorithm based on the prioritizing planning approach. This algorithm may use the path τ_1 already generated and plan a CT-path of $\mathcal{A}_2$ in $\mathcal{CT}_2$, treating $\mathcal{A}_1$ as a moving obstacle. If this algorithm fails again, a centralized planning algorithm may then be executed.

In general, the obstacle region in the $s_1 s_2$-space consists of several connected subsets whose shapes are arbitrarily complicated. Let us subdivide each path τ_i, $i = 1$ or 2, into a sequence of w_i path segments determined by the intervals $\delta_{i,k_i} = [s_{i,k_i}, s_{i,k_i+1}]$, with $k_i = 0, ..., w_i - 1$, $s_{i,0} = 0$ and $s_{i,w_i} = 1$. Typically, all the intervals $\delta_{i,k}$ along the same path τ_i determine subpaths of τ_i of equal length. This subdivision transforms the continuous $s_1 s_2$-space into an array of $w_1 \times w_2$ closed rectangular cells $\delta_{1,k_1} \times \delta_{2,k_2}$. In the following, we denote by κ_{k_1,k_2} the cell $\delta_{1,k_1} \times \delta_{2,k_2}$. Each cell κ_{k_1,k_2} is classified as EMPTY if the regions swept out by $\mathcal{A}_1$ and $\mathcal{A}_2$ when (s_1, s_2) varies in $\delta_{1,k_1} \times \delta_{2,k_2}$ do not intersect, i.e. if:

$$\{\mathcal{A}_1(\tau_1(s_1)) \,/\, s_1 \in \delta_{1,k_1}\} \cap \{\mathcal{A}_2(\tau_2(s_2)) \,/\, s_2 \in \delta_{2,k_2}\} = \emptyset.$$

The cell is classified as FULL otherwise. Hence, the two robots are guaranteed to not collide in an EMPTY cell (even in its boundary), while they may collide in a FULL cell. The classification of the cells may be more or less complicated depending on the shape of the two robots, but it is often reasonable to approximate the shape of the area swept out by a robot when its configuration varies in a path segment.

The discretized $s_1 s_2$-space with labeled cells is called a **coordination diagram**. Figure 5 shows an example of such a diagram, with EMPTY cells shown white. Let us restrict the possible paths in the coordination diagram to the network defined as follows. The nodes of the network

TESLA EXHIBIT 1013, Page 380

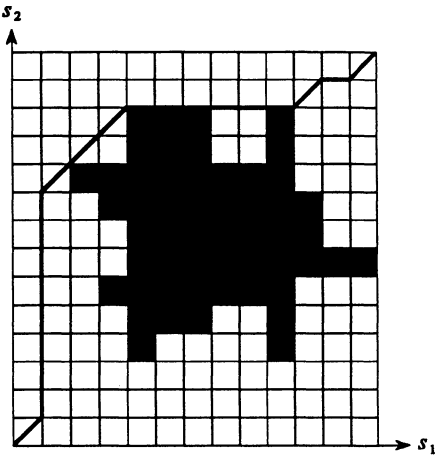


Figure 5. This figure shows the coordination diagram of two robots. Each rectangular cell determines segments of configurations of the two robots along two paths τ_1 and τ_2. Each cell is labeled as EMPTY (white cells) if the robots intersect with each other at none of these configurations; otherwise, it is labeled FULL (dark cells). The bold line shows a free schedule that is generated by searching the network formed by the edges bounding the EMPTY cells and their diagonals. A free schedule determines a class of coordinated motions along which the two robots do not collide with each other.

are all the vertices of the EMPTY cells. Its links are all the edges and the diagonals of these cells. We denote each node by its discretized coordinates (i, j), with $i \in [0, w_1]$ and $j \in [0, w_2]$. A free schedule can be generated by searching this network for a path connecting $(0, 0)$ to (w_1, w_2). The bold line in Figure 5 shows such a possible free schedule. (Remember that the boundary of an EMPTY cell is collision-free.)

We now describe a technique that allows us to generate a free schedule without search [O'Donnell and Lozano-Pérez, 1989]. This technique can only generate schedules that are nondecreasing with respect to both s_1 and s_2. It succeeds whenever one such free schedule exists. Prior to the description, we introduce the notion of SW-closure [Preparata and Shamos, 1985] [Yannakakis, Papadimitriou and Kung, 1979].

Let (s_1', s_2') and (s_1'', s_2'') be two points in the $s_1 s_2$-plane. They are **incomparable** if and only if $(s_1' - s_1'')(s_2' - s_2'') < 0$. Let us assume that

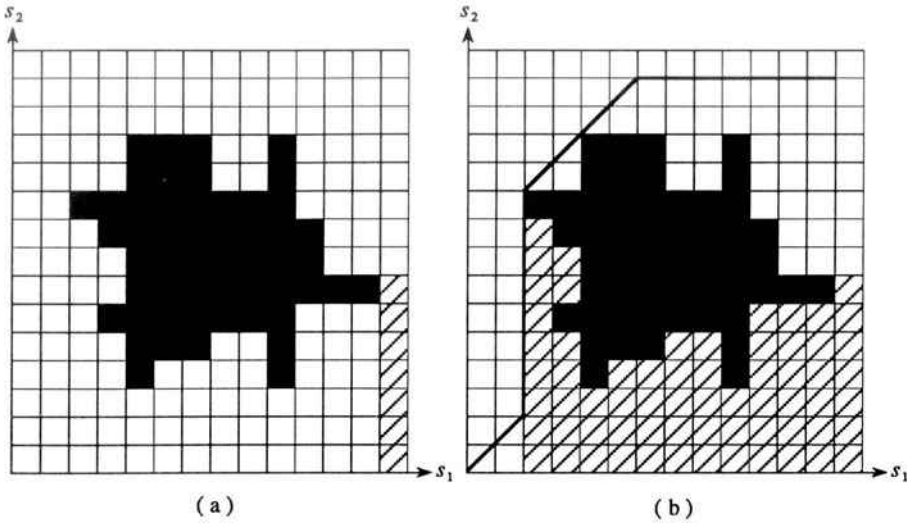


Figure 6. The extended coordination diagram is obtained by adding a fictitious path segment at the end of each of the two paths τ_1 and τ_2 (Figure a). Along this path segment the corresponding robot is motionless. The obstacle region is extended as shown in Figure a (striped cells). The SW-closure of the extended obstacle region is shown in Figure b (dark and striped cells). If there exists a nondecreasing free schedule, it only traverses cells that are not contained in the SW-closure of the extended obstacle region. The bold line in Figure b shows the free schedule constructed by the algorithm given in the text.

(s_1', s_2') and (s_1'', s_2'') are incomparable and (without loss of generality) that $s_1' < s_1''$ (hence: $s_2' > s_2''$). The **SW-conjugate** of (s_1', s_2') and (s_1'', s_2'') is the point (s_1', s_2''). A connected region R of the $s_1 s_2$-space is **SW-closed** if and only if the SW-conjugate of any two incomparable points in R is also in R. The **SW-closure** of a connected region S is the smallest SW-closed region R containing S. The SW-closure of a non-connected region is defined as the union of the SW-closures of all its connected components.

Now let us consider the coordination diagram. Let S be the obstacle region. We extend the diagram by adding a row and a column of cells at the top and the right of the diagram, respectively. This extension corresponds to adding a fictitious path segment at the end of each of the two paths τ_1 and τ_2. Along this path segment the corresponding robot

is motionless. We extend the region S as follows:

- If for any $j \in [0, w_2 - 1]$ the cell $\kappa_{w_1-1,j}$ is FULL, then all the cells $\kappa_{w_1,k}$, for $k = 0, ..., j$, are included in S.
- If for any $i \in [0, w_1 - 1]$ the cell κ_{i,w_2-1} is FULL, then all the cells κ_{k,w_2}, for $k = 0, ..., i$, are included in S.

Figure 6.a shows the extended diagram corresponding to the coordination diagram of Figure 5.

Let R be the SW-closure of S, as illustrated in Figure 6.b. It can be computed in time $O(w_1 w_2(\log w_1 + \log w_2))$ [Preparata and Shamos, 1985]. None of the nondecreasing free schedules in the coordination diagram traverse cells contained in R. In addition, a nondecreasing free schedule exists if and only if neither of the two cells $\kappa_{0,0}$ and κ_{w_1-1,w_2-1} are contained in R. More generally, a nondecreasing free schedule exists between (i, j), $0 \leq i \leq w_1$, $0 \leq j \leq w_2$, and (ω_1, ω_2) if and only if neither of the two cells $\kappa_{i,j}$ and κ_{w_1-1,w_2-1} are contained in R.

Therefore, if a nondecreasing free schedule exists, one such schedule is constructed (without searching any graph) by the following algorithm:

```
i ← 0; j ← 0;
while i < w1 or j < w2 do
    begin
        if κi,j ∉ R then
            if i < w1 and j < w2 then
                    i ← i + 1; j ← j + 1; move to (i, j);
            else
                if i < w1 then
                        i ← i + 1; move to (i, j);
                else
                        j ← j + 1; move to (i, j);
        else
            if i < w1 and κi,j−1 ∉ R then
                    i ← i + 1; move to (i, j);
            else
                    j ← j + 1; move to (i, j);
    end;
```

A free schedule constructed by this algorithm is shown as a bold line in Figure 6.b.

3 Articulated Robots

In this section we consider the case where the robot is made of several moving rigid objects connected by mechanical joints which constrain their relative motion. A manipulator arm is a typical example of such a robot. It consists of a sequence of rigid objects connected in a chain by joints.

3.1 Configuration Space

Consider a robot $\mathcal{A}$ made of p rigid objects $\mathcal{A}_1, ..., \mathcal{A}_p$. Any two objects $\mathcal{A}_i$ and $\mathcal{A}_j$ may possibly be connected by a joint. For simplification, we will assume that the joint is either a **revolute joint** or a **prismatic joint**. A revolute joint is a hinge that constrains the relative motion of $\mathcal{A}_i$ and $\mathcal{A}_j$ to be a rotation around an axis fixed with respect to both objects. A prismatic joint is a sliding connection that constrains the relative motion to be a translation along an axis fixed with respect to both objects. These are the most usual joints, but other types of joint are possible. Many of them would be treated in much the same way as we will treat the revolute and prismatic joints.

Let $\mathcal{F}_{\mathcal{A}_i}$ be the frame attached to $\mathcal{A}_i$, $i \in [1, p]$. As defined in Section 2, the configuration of $\mathcal{A}$ is a specification of the position and orientation of every frame $\mathcal{F}_{\mathcal{A}_i}$, $i = 1$ to p, with respect to $\mathcal{F}_{\mathcal{W}}$. If the various objects were independent free-flying objects in $\mathcal{W} = \mathbf{R}^N$, the configuration space of $\mathcal{A}$ would be:

$$\mathcal{C}' = \underbrace{(\mathbf{R}^N \times SO(N)) \times ... \times (\mathbf{R}^N \times SO(N))}_{p}.$$

However, the various joints impose constraints[6] on the feasible configurations in $\mathcal{C}'$. These constraints select out a subspace $\mathcal{C}$ of $\mathcal{C}'$ of lower dimension which is the actual configuration space of $\mathcal{A}$.

[6] In the next chapter, each such constraint will be called a *holonomic* kinematic constraint.

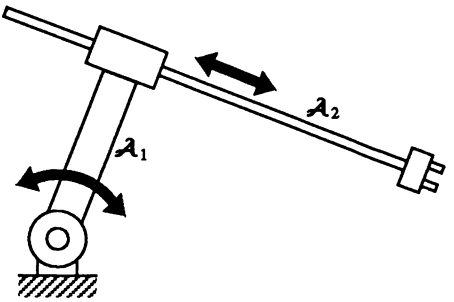


Figure 7. This planar manipulator arm $\mathcal{A}$ consists of two objects $\mathcal{A}_1$ and $\mathcal{A}_2$. $\mathcal{A}_1$ is connected to the workspace by a revolute joint, and $\mathcal{A}_2$ to $\mathcal{A}_1$ by a prismatic joint. The configuration space of $\mathcal{A}$ is a two-dimensional subspace of $(\mathbf{R}^2 \times SO(2))^2$.

Example: Consider the planar two-joint manipulator arm shown in Figure 7. It consists of two objects $\mathcal{A}_1$ and $\mathcal{A}_2$. $\mathcal{A}_1$ is connected to the workspace by a revolute joint; $\mathcal{A}_2$ is connected to $\mathcal{A}_1$ by a prismatic joint. The workspace is $\mathcal{W} = \mathbf{R}^2$. In this example, $\mathcal{C}' = (\mathbf{R}^2 \times SO(2))^2$ is a six-dimensional space. The first joint imposes two constraints expressing that the point about which $\mathcal{A}_1$ can rotate is fixed in the workspace. The second joint imposes two additional constraints on the position and orientation of $\mathcal{A}_2$ relative to $\mathcal{A}_1$. The four constraints are independent and determine a subspace $\mathcal{C}$ of dimension 2 in $\mathcal{C}'$. ∎

It is in general suitable to consider $\mathcal{C}$ independently of $\mathcal{C}'$, since its dimension is usually much smaller.

In the rest of this section, we assume that $\mathcal{A}$ is an articulated robot containing no kinematic loop, i.e. no closed sequence $\mathcal{A}'_1, ..., \mathcal{A}'_r, \mathcal{A}'_{r+1}$, with $\mathcal{A}'_{r+1} = \mathcal{A}'_1$ and $\mathcal{A}'_{i+1}$ being connected to $\mathcal{A}'_i$ by a joint, for every $i \in [1, r]$. Then the total number of joints in $\mathcal{A}$ is exactly p.

Under the previous assumption, let us consider the workspace as a fictitious object denoted $\mathcal{A}_0$. Every object $\mathcal{A}_j$ ($j \in [1, p]$) is connected to a *single* object $\mathcal{A}_i$, $i \in [0, p]$, $i \neq j$. Hence, the whole robot $\mathcal{A}$ can be represented by a tree structure, whose nodes are the objects $\mathcal{A}_0$ through $\mathcal{A}_p$ and whose arcs are the joints. The root of the tree is $\mathcal{A}_0$. For any two objects $\mathcal{A}_i$ and $\mathcal{A}_j$, such that $\mathcal{A}_j$ is a child of $\mathcal{A}_i$ in the tree, we can define the configuration of $\mathcal{A}_j$ relative to $\mathcal{A}_i$ as a specification of the

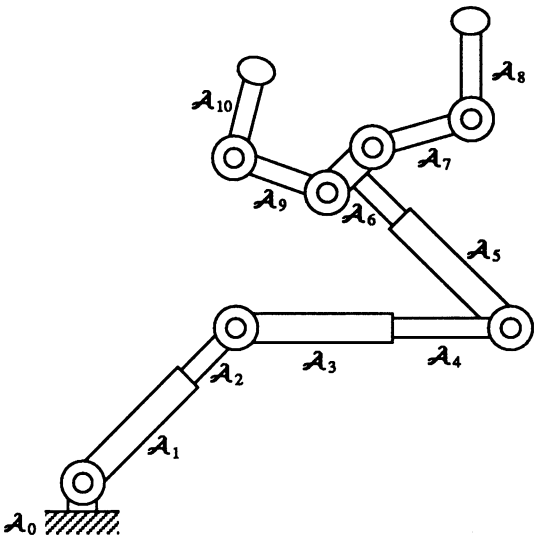


Figure 8. This figure depicts an articulated robot with three telescopic prismatic joints and seven revolute joints. Assuming no mechanical stops in the joints, the configuration space of this robot is $\mathcal{C} = (S^1)^7 \times \mathbf{R}^3$.

position and orientation of $\mathcal{F}_{\mathcal{A}_j}$ with respect to $\mathcal{F}_{\mathcal{A}_i}$. Let $\mathcal{C}_j^{(i)}$ denote the configuration space of $\mathcal{A}_j$ relative to $\mathcal{A}_i$. Assume for a moment that there is no mechanical stop in the joints. If the two objects are connected by a revolute joint, we have that: $\mathcal{C}_j^{(i)} = S^1$. If the two objects are connected by a prismatic joint, we have that: $\mathcal{C}_j^{(i)} = \mathbf{R}$. It follows that the configuration space $\mathcal{C}$ of an articulated object $\mathcal{A}$ with no loop, made of p rigid objects connected by p_1 revolute joints and p_2 prismatic joints ($p_1 + p_2 = p$), is [Burdick, 1988]:

$$\mathcal{C} = \underbrace{S^1 \times \ldots \times S^1}_{p_1} \times \underbrace{\mathbf{R} \times \ldots \times \mathbf{R}}_{p_2}.$$

Hence, the configuration space of the two-joint manipulator arm of Figure 7 is $S^1 \times \mathbf{R}$. In the same way, the configuration space of the ten-joint robot shown in Figure 8 is $(S^1)^7 \times \mathbf{R}^3$.

Therefore, the configuration space of an articulated robot with p revolute and prismatic joints and no kinematic loop is a p-dimensional smooth manifold. A chart of this manifold can be simply defined by associating

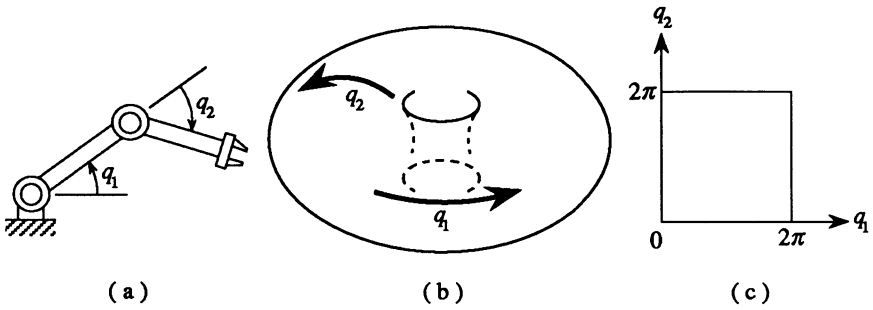


Figure 9. The configuration space of a planar two-revolute-joint manipulator arm (Figure a) is a torus in $\mathbf{R}^3$ (Figure b). A parameterization of this space is obtained by representing every configuration as $(q_1, q_2) \in [0, 2\pi) \times [0, 2\pi)$ (see Figure c), where q_1 and q_2 are angles associated with the joints as illustrated in Figure a. This parameterization corresponds to cutting the torus along two generators. Opposite edges of the square shown in Figure c must be procedurally identified through modulo 2π arithmetic in order to capture the multiple connectedness of the torus.

an angle, say in $(0, 2\pi)$, with each revolute joint and a real number with each prismatic joint. Then, a configuration $\mathbf{q}$ is represented by a list $(q_1, ..., q_p)$ of coordinates, each one describing the relative position and orientation of two joint-connected objects in $\mathcal{A}$. Robots with revolute joints have a multiply-connected configuration space and several charts are necessary to form an atlas. However, in practice, it is often sufficient to consider a single chart, with each angular coordinate belonging to, say, $[0, 2\pi)$, and applying modulo 2π arithmetic to the coordinate.

Example: Consider the planar two-revolute-joint manipulator arm of Figure 9.a. The configuration space of this robot is $S^1 \times S^1$, hence a torus in a three-dimensional Euclidean space (Figure 9.b). A chart on this manifold can be defined by associating an angle in $(0, 2\pi)$ with each of the two joints. Figure 9.a illustrates two such angles q_1 and q_2. The torus is multiply-connected and several charts are required to form an atlas. Nevertheless, in most cases, it is appropriate to consider a single chart and let the two angles q_1 and q_2 vary in $[0, 2\pi)$, with modulo 2π arithmetic. This simplification corresponds to representing the torus by a square $[0, 2\pi] \times [0, 2\pi]$ (see Figure 9.c) whose opposite edges are procedurally identified. ∎

TESLA EXHIBIT 1013, Page 387

The notions of **C-obstacle**, **free space** and **free path** defined for multiple objects extend to an articulated robot without modification. There are also two sorts of C-obstacles: those due to the interaction of an object $\mathcal{A}_i$ and an obstacle $\mathcal{B}_j$, and those due to the interaction between two objects $\mathcal{A}_i$ and $\mathcal{A}_j$. Figure 10 illustrates the construction of C-obstacles of the first sort for a two-revolute-joint planar manipulator. The computation of the exact equations of a C-obstacle boundary for an articulated robot has been recently investigated in [Hwang, 1990], [Dooley and McCarthy, 1990] and [Ge and McCarthy, 1990].

In general, the relative motion of two objects connected by a joint is further limited by a pair of *mechanical stops*, which restrict the range of values of the corresponding configuration coordinate to a connected interval. The mechanical stops thus act like "invisible obstacles". The set of configurations where no joint has reached a mechanical stop is an open subset of $\mathcal{C}$. The complement of this subset is the "C-obstacle region" to which the invisible obstacles due to the mechanical stops map. The mechanical stops in every joint are usually independent of the configuration of the robot. Then the subset of configurations that are left feasible by the mechanical stops is simply a cross-product of intervals in the Cartesian space (chart) representing $\mathcal{C}$.

Consider a revolute joint connecting $\mathcal{A}_i$ to $\mathcal{A}_j$. If the mechanical stops in this joint prevent $\mathcal{A}_j$ from making a full rotation of 2π with respect to $\mathcal{A}_i$, the limitation eliminates the need for applying modulo 2π arithmetic to the corresponding configuration coordinate. If, instead, the mechanical stops allow $\mathcal{A}_i$ to perform r ($r > 1$) full rotations[7], then it is appropriate to let the corresponding configuration coordinate vary in an interval of length $2r\pi$ (r does not have to be an integer). In this way, the configuration is explicitly represented with respect to both mechanical stops. One drawback of this representation, however, is that the same obstacle may map to several C-obstacle regions of identical shape.

3.2 Path Planning Methods

The configuration space $\mathcal{C}$ of an articulated robot is just another smooth manifold. As long as the constraints imposed by the joints on the relative

[7]This is often the case with some revolute joints located in the "wrist" of a manipulator arm.

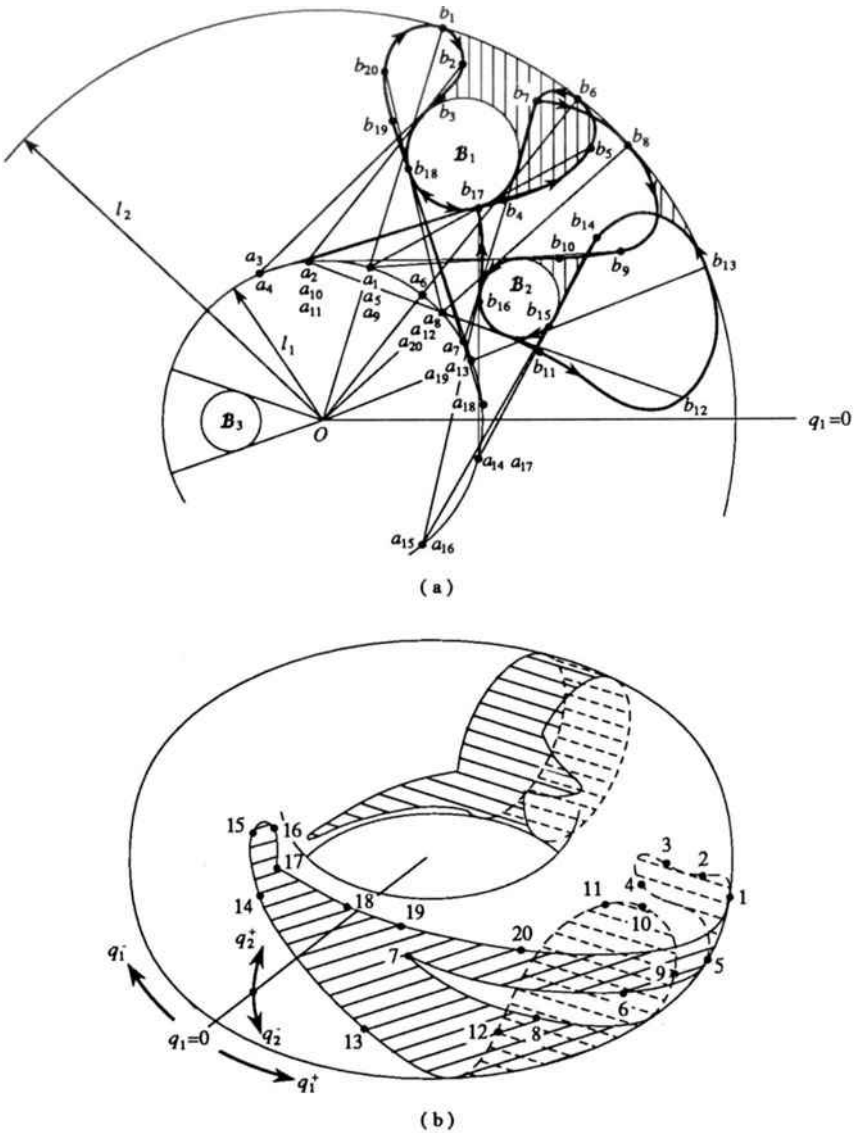


Figure 10. The robot is a two-revolute-joint planar manipulator moving among three circular obstacles $\mathcal{B}_1$, $\mathcal{B}_2$ and $\mathcal{B}_3$ (Figure a). Its two links are line segments of lengths l_1 and l_2. The first joint is located at O. The second moves in the circle of radius l_1 centered at O. The configuration space is the torus $S^1 \times S^1$ shown in Figure b. The striped areas in Figure b depict the C-obstacles. This construction is due to Lumelsky [Lumelsky, 1987b].

TESLA EXHIBIT 1013, Page 389

motion of the objects they connect can be expressed in algebraic form (which is the case with revolute and prismatic joints) the general methods described in Section 4 of Chapter 4 (silhouette method) and in Section 3 of Chapter 5 (Collins decomposition) are applicable. These methods take simple and double exponential time, respectively, in the dimension of the configuration space $\mathcal{C}$.

An extension of the freeway method presented in Section 3 of Chapter 4 is described in [Brooks, 1983b] for a specific manipulator arm with revolute joints only.

The approximate cell decomposition approach is also applicable, although the mapping of the obstacles in the configuration space is more difficult than for the basic motion planning problem with polygonal/polyhedral objects. We study this mapping in detail in the next subsection, in the case of a manipulator arm. Several path planners based upon the approximate cell decomposition approach have been implemented for manipulator arms (e.g. [Gouzènes, 1984b] [Laugier and Germain, 1985] [Faverjon, 1986] [Lozano-Pérez, 1987]). However, the approach is limited to robots with relatively few joints.

The potential field approach can be applied to articulated objects with minor adaptations. These adaptations are presented below in Subsection 3.4. So far, the best practical results for robots with many joints have been obtained with this approach. For instance, the randomized planner described in Section 5 of Chapter 7 has been successfully experimented with problems involving up to a few tens of joints in two- and three-dimensional workspace [Barraquand and Latombe, 1989a]. The learning technique introduced in Section 6 of Chapter 7 has been successfully applied to plan the coordinated motion of two six-joint manipulator arms [Faverjon and Tournassoud, 1989].

Because the motions of the rigid objects in an articulated robot are strongly interdependent, the "decoupled planning" techniques described in Subsection 2.3 do not apply.

3.3 Approximate Cell Decomposition

The principle of the approximate cell decomposition approach is general enough to be applicable to an articulated robot. The main issue is the construction of both the free space and the C-obstacle region in

the Cartesian space representing $\mathcal{C}$. In Chapter 6 we used the explicit equations defining the boundaries of the C-obstacles to label the rectangloid cells generated by the decomposition of the configuration space. These equations were established in Chapter 3 for a polygonal/polyhedral object moving freely among polygonal/polyhedral obstacles. We could use a similar approach here, but obviously, the explicit equations of the boundaries of the C-obstacles are now considerably more difficult to establish and to exploit. Instead, we describe a method that generates a conservative approximation of the free space (i.e. a subset of the actual free space) by discretizing the motion of each joint into small intervals. This approximate representation is then used to construct a connectivity graph, which is searched for a channel connecting the initial configuration to the goal configuration. With minor variations, this is the method proposed in [Gouzènes, 1984b], [Laugier and Germain, 1985], [Faverjon, 1986] and [Lozano-Pérez, 1987]. Various implementations and experimental results of the method are presented in these publications.

In order to simplify our presentation, we make a few assumptions which are not difficult to remove. First, we assume that $\mathcal{A}$ is a sequence of p objects $\mathcal{A}_1$ through $\mathcal{A}_p$, with every two successive objects $\mathcal{A}_i$ and $\mathcal{A}_{i+1}$ being connected by either a revolute or a prismatic joint. $\mathcal{A}_1$ is the first object in the sequence and is connected to the workspace by a joint. As introduced in Subsection 3.1, we represent a configuration $\mathbf{q}$ of $\mathcal{A}$ by a list of p parameters, $(q_1, ..., q_p)$. Each parameter q_i $(i \in [1,p])$ determines the position and orientation of $\mathcal{F}_{\mathcal{A}_i}$ relative to $\mathcal{F}_{\mathcal{A}_{i-1}}$, with $\mathcal{F}_{\mathcal{A}_0} = \mathcal{F}_{\mathcal{W}}$. The second assumption is that each q_i varies in a bounded interval $I_i = (q_i^-, q_i^+)$. If q_i corresponds to a revolute joint with no mechanical stop, the interval is $I_i = [0, 2\pi)$ and modulo 2π arithmetic is used. The third assumption is that no two objects $\mathcal{A}_i$ and $\mathcal{A}_j$ can ever collide with each other. Hence, the only collisions we consider are those between the objects $\mathcal{A}_i$ $(i \in [1,p])$ and the obstacles $\mathcal{B}_j$ $(j \in [1,q])$.

The position and orientation of $\mathcal{F}_{\mathcal{A}_i}$ relative to $\mathcal{F}_{\mathcal{W}}$, for any $i \in [1,p]$, is completely determined by the parameters q_1 through q_i. Hence, we denote by $\mathcal{A}_i(q_1, ..., q_i)$ the region of $\mathcal{W}$ occupied by the object $\mathcal{A}_i$.

It is possible to construct an approximation of the C-obstacle region in $I_1 \times ... \times I_p \subset \mathbf{R}^p$ as follows. Each interval I_i, $i = 1$ to p, is partitioned into non-overlapping smaller intervals δ_{i,k_i}, $k_i = 1, 2, ...$, at some prespecified resolution. If the region swept out by $\mathcal{A}$ in the workspace,

TESLA EXHIBIT 1013, Page 391

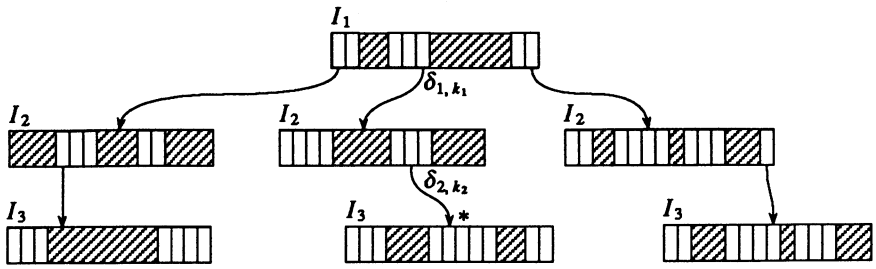


Figure 11. The configuration space of a manipulator arm is represented in the form of a p-level tree structure [Lozano-Pérez, 1987]. The nodes at depth $i-1$ ($i \in [1,p]$) represent decompositions of the range I_i into EMPTY and FULL intervals. Except at depth $p-1$, the EMPTY intervals are decomposed into smaller intervals δ_{i,k_i}. Consider the node at depth 2 marked by "*". The EMPTY intervals of this node are formed by all the values of q_3 such that $\mathcal{A}_3(q_1, q_2, q_3)$ intersects with no obstacle when q_1 varies over δ_{1,k_1} and q_2 varies over δ_{2,k_2}.

when $(q_1, ..., q_p)$ varies over the rectangloid cell $\delta_{1,k_1} \times ... \times \delta_{p,k_p}$, does not intersect with any obstacle, then $\delta_{1,k_1} \times ... \times \delta_{p,k_p}$ is part of the free space $\mathcal{C}_{free}$; otherwise, it is considered to be in the C-obstacle region. By considering all the cells, we construct a conservative approximation of the free space as a collection of small rectangloid cells. The connectivity graph representing the adjacency relation among these cells can then be generated and searched for a channel. As such, however, the overall planning method is not likely to be very efficient in terms of computing time and memory space. Fortunately, in most cases, simple modifications allow for a considerable reduction in the number of cells that have to be explicitly checked by this method. These modifications are described below.

Let us consider the first object $\mathcal{A}_1$ in $\mathcal{A}$. Its position and orientation in $\mathcal{W}$ depend only on the value of the coordinate q_1. If, for a given value of this coordinate, it intersects with an obstacle, then there is no need to check $\mathcal{A}_2$ through $\mathcal{A}_p$ for collisions with obstacles. Indeed, the whole $(p-1)$-dimensional slice of $\mathcal{C}$ projecting at the given value of q_1 is part of the C-obstacle region. In the same way, if for given values of $q_1, ..., q_i$, $\mathcal{A}_i$ intersects with an obstacle, the whole $(p-i)$-dimensional slice of $\mathcal{C}$ projecting at the given values of q_1 through q_i is part of the C-obstacle

region. This suggests constructing a representation of the configuration space in the form of a p-level tree, as shown in Figure 11 [Gouzènes, 1984b] [Lozano-Pérez, 1987].

Each node at depth $i-1$ ($i \in [1,p]$) in this tree represents a decomposition of the range I_i of values of q_i into intervals classified as EMPTY or FULL. The root of the tree, i.e. the node at depth 0, represents the range I_1. The (open) EMPTY intervals of the decomposition are formed by all the values of q_1 for which $\mathcal{A}_1(q_1)$ intersects with no obstacle. The (closed) FULL intervals are the complement of the EMPTY intervals in I_1. The EMPTY intervals are subdivided into smaller intervals δ_{1,k_1}, $k_1 = 1, 2, ...$, at the specified resolution. The tree contains one node at depth 1 for each such interval. Consider any interval δ_{1,k_1}. The corresponding successor represents a decomposition of the range I_2 into EMPTY and FULL intervals. The EMPTY intervals are formed by all the values of q_2 for which $\mathcal{A}_2(q_1, q_2)$ intersects with no obstacle when q_1 varies over δ_{1,k_1}. In turn, these intervals are decomposed into smaller intervals δ_{2,k_2}, and so on. Since entire slices may get classified as part of the C-obstacle region at early stages of the computation, the time and memory required to build the tree structure is usually much smaller than with the more simplistic version of the method presented above.

When the tree is fully computed, we have an approximate representation of the free space and the C-obstacle region. In particular, given a configuration $(q_1, ..., q_p)$, we can easily check whether it belongs to the approximate free space or not. Indeed, the configuration determines a unique path in the tree. If this path terminates at a FULL interval (this may happen at any depth in the tree), the configuration is considered to be in the C-obstacle region. Instead, if the path terminates in an EMPTY interval (this can only happen at depth $p-1$), the configuration is in free space.

Figure 12 illustrates the representation of the configuration space of a two-revolute-joint arm computed by the above algorithm. The C-obstacle region (shown dark in the figure) is approximated as a set of slices parallel to the q_2-axis.

The key computation for constructing the tree of Figure 11 is to determine the range of values of q_i ($i \in [1,p]$) at which $\mathcal{A}_i(q_1, ..., q_{i-1}, q_i)$ intersects no obstacle when $q_1, ..., q_{i-1}$ vary over intervals $\delta_{1,k_1}, ..., \delta_{i-1,k_{i-1}}$.

TESLA EXHIBIT 1013, Page 393

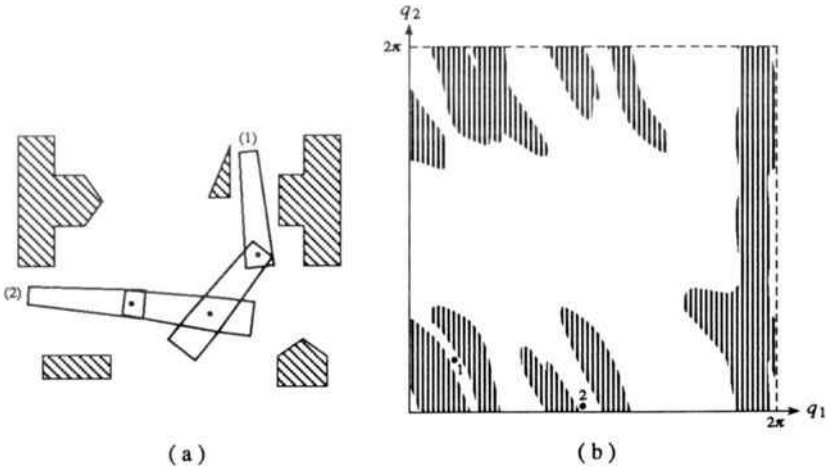


Figure 12. Figure a shows a planar two-revolute-joint arm among obstacles, at two free configurations marked (1) and (2). We assume that there is no mechanical stop in the joints. Figure b shows the representation of the robot's configuration space constructed by the algorithm described in the text. The C-obstacle region (shown dark) is approximated by a set of slices parallel to the q_2-axis. The points designated by 1 and 2 represent the arm's configurations marked (1) and (2) in Figure a.

A simple technique proposed in [Lozano-Pérez, 1987] consists of fixing the values of $q_1, ..., q_{i-1}$ to, say, the midpoints $q_{1,k_1}, ..., q_{i-1,k_{i-1}}$ of the intervals $\delta_{1,k_1}, ..., \delta_{i-1,k_{i-1}}$. Then the computation reduces to determining the intervals of values of q_i for which $\mathcal{A}_i(q_{1,k_1}, ...q_{i-1,k_{i-1}}, q_i)$ intersects no obstacle when it rotates around a fixed axis (in the case of a revolute joint) or translates along a fixed axis (in the case of a prismatic joint). If $\mathcal{A}_i$ and the obstacles are modeled by polygons or polyhedral, this computation is not difficult. (The case where $\mathcal{A}_i$ is a polygon rotating about a fixed point among polygonal obstacles has already been studied in Subsection 5.2 of Chapter 6.)

However, fixing the values of q_1,...,q_{i-1}, rather than letting them vary over intervals, has the drawback of generating an approximation of the free space that is no longer guaranteed to be a conservative one. One way to deal with this drawback is to compute the largest Euclidean displacement ρ_i of any point in $\mathcal{A}_i$ for any $q_i \in I_i$, when $q_1, ..., q_{i-1}$ vary over $\delta_{1,k_1}, ..., \delta_{i-1,k_{i-1}}$. The object $\mathcal{A}_i$ is grown isotropically by ρ_i and

TESLA EXHIBIT 1013, Page 394

the grown object (rather than $\mathcal{A}_i$) is used to decompose the range I_i into EMPTY and FULL intervals.

The construction of the tree representing configuration space can be improved further as follows [Gouzènes, 1984b] [Faverjon, 1986]. Let $\mathcal{S}_i(q_1, ..., q_i)$, $i \in [1, p]$, be the region swept out by the objects $\mathcal{A}_{i+1}, ..., \mathcal{A}_p$ when $q_{i+1}, ..., q_p$ vary over $I_{i+1}, ..., I_p$, and the previous i objects $\mathcal{A}_1, ..., \mathcal{A}_i$ are at configuration $(q_1, ..., q_i)$. Note that only the position and orientation of $\mathcal{S}_i(q_1, ..., q_i)$, not its shape, depend on the values of $q_1, ..., q_i$. When the range I_i labeling a node at depth $i-1$ in the tree is decomposed, the generated intervals are labeled as FULL, C/EMPTY (for "certainly empty") and P/EMPTY (for "possibly empty"):

- The FULL intervals are the same as before, i.e. they are formed by all the values of q_i such that $\mathcal{A}_i(q_1, ..., q_{i-1}, q_i)$ intersects with an obstacle when $q_1, ..., q_{i-1}$ vary over $\delta_{1,k_1}, ..., \delta_{i-1,k_{i-1}}$.
- The C/EMPTY intervals are formed by all the values of q_i such that $\mathcal{A}_i(q_1, ..., q_{i-1}, q_i) \cup \mathcal{S}_i(q_1, ..., q_{i-1}, q_i)$ intersects with no obstacle when $q_1, ..., q_{i-1}$ vary over $\delta_{1,k_1}, ..., \delta_{i-1,k_{i-1}}$.
- The P/EMPTY intervals are the complementary of the FULL and C/EMPTY intervals in I_i.

Only the P/EMPTY intervals of I_i are subdivided into smaller intervals δ_{i,k_i}. With each P/EMPTY interval in a node of the tree, one can associate the list of obstacles with which $\mathcal{S}_i$ collides. When the successors of this node (corresponding to this interval) are considered, only the obstacles in this list have to be considered for checking the possible collisions of $\mathcal{A}_{i+1}$. Then, in each P/EMPTY interval of I_{i+1}, if any, the list is pruned further by removing the obstacles that do not collide with $\mathcal{S}_{i+1}$ [Faverjon and Tournassoud, 1989].

Once the tree representation of the robot's configuration space has been built, we still have the problem of constructing a free path between the initial configuration and the goal one. The tree directly provides a representation of the free space in the form of a collection of rectangloid cells. The connectivity graph representing the adjacency relation between these cells can be constructed and searched for a channel. However, the cells derived from the tree representation are very thin along one or several dimensions and usually there are many of them, resulting

in a large connectivity graph. It may be preferable to group these cells into larger rectangloids. One way[8] to do this is to consider the tree representation as the exact representation of the configuration space and to apply an approximate cell decomposition method (as described in Section 1 of Chapter 6) to this representation. Using the information contained in the tree, the labeling of the new cells generated by this method as EMPTY, FULL and MIXED is particularly simple.

3.4 Potential Field

The potential field approach described in Chapter 7 applies to articulated robots. As is the case for a single rigid object, there are many possible methods based on this approach.

Typically, one considers a collection of control points $a_j \in \mathcal{A}$, $j = 1, ..., Q$, which may be distributed over the various objects $\mathcal{A}_i$. Each point a_j is subject to a potential field $\mathbf{V}_j(\mathbf{x})$ defined in the workspace. This potential may be either an attractive potential depending on the goal position of a_j in $\mathcal{W}$ (see Subsection 2.2, Chapter 7), a repulsive potential depending on the obstacles in the workspace (see Subsection 2.3, Chapter 7), or a mixed potential depending on both the goal position of a_j and the obstacles (see Subsection 4.2.3, Chapter 7).

The potential function $\mathbf{U}(\mathbf{q})$ in the robot's configuration space is obtained by adding these potentials:

$$\mathbf{U}(\mathbf{q}) = \sum_{j=1}^{Q} \mathbf{V}_j(a_j(\mathbf{q})).$$

It induces the attractive force defined by:

$$\vec{F}(\mathbf{q}) = -\vec{\nabla}\mathbf{U}(\mathbf{q}).$$

Let us consider a chart in the robot's configuration space $\mathcal{C}$ such that a configuration $\mathbf{q}$ is parameterized by $(q_1, ..., q_p)$, with each q_i defining the relative position of two objects connected by a joint, as described in Subsection 3.1. The components of $\vec{F}(\mathbf{q})$ in the basis of $T_{\mathbf{q}}(\mathcal{C})$ (tangent space of $\mathcal{C}$ at $\mathbf{q}$) induced by this chart are the forces/torques applied at every prismatic/revolute joint of the robot. Assume further that $\mathcal{W} =$

[8] See [Lozano-Pérez, 1987] [Siméon, 1989] for other techniques.

TESLA EXHIBIT 1013, Page 396

$\mathbf{R}^3$, with x, y, z being the coordinates of a point $\mathbf{x} \in \mathcal{W}$ in $\mathcal{F}_{\mathcal{W}}$. Let $X_j(q_1, ..., q_p)$, $Y_j(q_1, ..., q_p)$ and $Z_j(q_1, ..., q_p)$ be the coordinates of $a_j(\mathbf{q})$ in $\mathcal{F}_{\mathcal{W}}$. The ith component of $\vec{F}(\mathbf{q})$ is:

$$-\frac{\partial}{\partial q_i}\mathbf{U}(q_1, ..., q_p) = -\sum_{j=1}^{Q}\left(\frac{\partial \mathbf{V}_j}{\partial x}\frac{\partial X_j}{\partial q_i} + \frac{\partial \mathbf{V}_j}{\partial y}\frac{\partial Y_j}{\partial q_i} + \frac{\partial \mathbf{V}_j}{\partial z}\frac{\partial Z_j}{\partial q_i}\right).$$

Hence, we have that:

$$\vec{F}(\mathbf{q}) = -\sum_{j=1}^{Q} J_j^T(\mathbf{q})\vec{\nabla}\mathbf{V}_j(\mathbf{x})_{|\mathbf{x}=a_j(\mathbf{q})}$$

where J_j^T is the transpose of the Jacobian matrix J_j of the map:

$$(q_1, ..., q_p) \in \mathbf{R}^p \mapsto (X_j(q_1, ..., q_p), Y_j(q_1, ..., q_p), Z_j(q_1, ..., q_p)) \in \mathbf{R}^3.$$

The same result also holds with $\mathcal{W} = \mathbf{R}^2$.

If the motion at a joint is limited by mechanical stops, these stops can be taken into account in the potential function. One way to proceed is as follows [Khatib, 1986]. Let a configuration coordinate q_i be constrained to be in an interval (q_i^-, q_i^+) by the mechanical stops in the corresponding joint. Both endpoints of the interval can be treated as obstacles generating repulsive potential fields $\mathbf{W}^-(q_i)$ and $\mathbf{W}^+(q_i)$. Using the same general form as for the repulsive potential functions described in Subsection 2.3 of Chapter 7, we define $\mathbf{W}^-(q_i)$ and $\mathbf{W}^+(q_i)$ by:

$$\mathbf{W}^-(q_i) = \begin{cases} \frac{1}{2}\eta\left(\frac{1}{q_i - q_i^-} - \frac{1}{\rho_0}\right)^2 & \text{if } q_i - q_i^- \leq \rho_0, \\ 0 & \text{if } q_i - q_i^- > \rho_0 \end{cases}$$

and:

$$\mathbf{W}^+(q_i) = \begin{cases} \frac{1}{2}\eta\left(\frac{1}{q_i^+ - q_i} - \frac{1}{\rho_0}\right)^2 & \text{if } q_i^+ - q_i \leq \rho_0, \\ 0 & \text{if } q_i^+ - q_i > \rho_0. \end{cases}$$

where η is a positive scaling factor and ρ_0 is the "distance of influence" of the mechanical stops. The various functions $\mathbf{W}_i^-(q_i)$ and $\mathbf{W}_i^+(q_i)$ are added to the function $\mathbf{U}(q_1, ..., q_p)$ in order to get a potential function

TESLA EXHIBIT 1013, Page 397

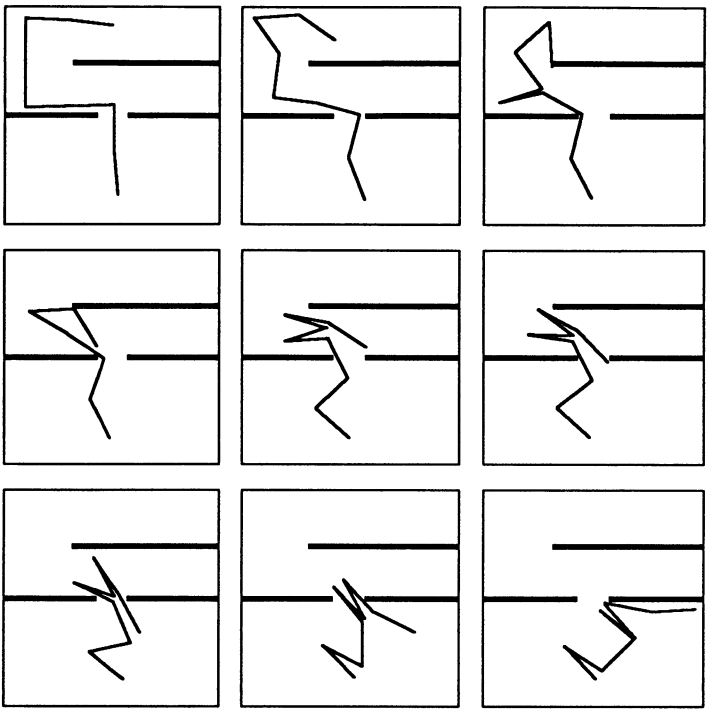


Figure 13. This figure shows snapshots along a path generated for an eight-revolute-joint robot by an implementation of the randomized potential-field-based method described in Section 5 of Chapter 7.

that takes the mechanical stops into account. Notice that all the components of the forces induced by $\mathbf{W}_i^-(q_i)$ and $\mathbf{W}_i^+(q_i)$, except for the ith ones, are zero. The ith components of these forces are:

$$-\frac{d\mathbf{W}_i^-(q_i)}{dq_i} = \begin{cases} \eta(\frac{1}{q_i - q_i^-} - \frac{1}{\rho_0}) & \text{if } q_i - q_i^- \leq \rho_0, \\ 0 & \text{if } q_i - q_i^- > \rho_0 \end{cases}$$

and:

$$-\frac{d\mathbf{W}_i^+(q_i)}{dq_i} = \begin{cases} \eta(\frac{1}{q_i^+ - q_i} - \frac{1}{\rho_0}) & \text{if } q_i^+ - q_i \leq \rho_0, \\ 0 & \text{if } q_i^+ - q_i > \rho_0. \end{cases}$$

These are forces/torques applying to the ith joint of the robot.

TESLA EXHIBIT 1013, Page 398

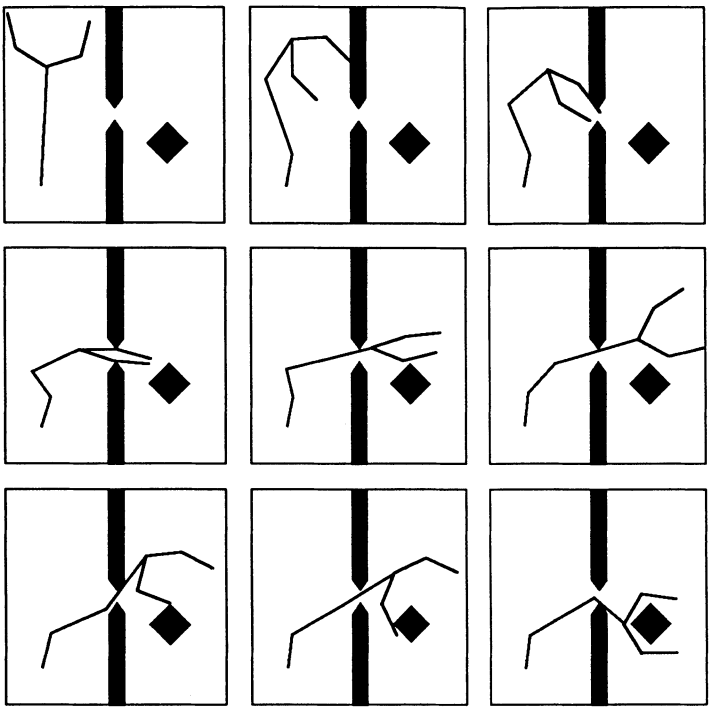


Figure 14. This figure shows snapshots along a path generated for a ten-joint robot (see Figure 8) by an implementation of the randomized potential-field-based method described in Section 5 of Chapter 7.

The various potential field methods described in Chapter 7 can be used to plan free paths for manipulator arms. As mentioned in Subsection 3.2, several potential-field-based planners have been implemented. As an illustration of the power of this approach, Figures 13 and 14 show snapshots along two paths generated by the randomized planner described in Section 5 of Chapter 7 [Barraquand, Langlois and Latombe, 1989b]. Figure 13 illustrates a path for an eight-revolute-joint robot. Figure 14 illustrates a path for the ten-joint robot of Figure 8. The example of Figure 13 was produced using a single control point located at the endpoint of the arm. The example of Figure 14 was produced using two control points located at the endpoints of the two kinematic chains. In both examples, collisions of the various links among themselves were checked explicitly in the same way as the collisions between the robot

and the obstacles. Mechanical stops were imposed on each joint and the collisions with each stop were also checked.

Exercises

1: (One-dimensional asteroid problem) Let $\mathcal{A}$ be a polygonal object moving in translation along a single line L (locus of the origin of the frame $\mathcal{F}_\mathcal{A}$ attached to $\mathcal{A}$) in the plane $\mathbf{R}^2$. The plane contains moving obstacles $\mathcal{B}_i$, $i = 1, ..., q$. Each obstacle $\mathcal{B}_i$ is a convex polygon translating without rotation at constant linear velocity $\vec{\mathbf{v}}_{\mathcal{B}_i}$. No two obstacles ever collide. The geometry, the position at time 0, and the velocity of each obstacle are given. The problem is to plan a motion of $\mathcal{A}$ from an initial position x_{init} in L at time 0 to a goal position x_{goal} at time T. The velocity $\vec{\mathbf{v}}_\mathcal{A}$ of $\mathcal{A}$ is bounded, i.e. $\|\vec{\mathbf{v}}_\mathcal{A}\| \leq V_{max}$. There is no constraint on its acceleration.

a. Characterize the shape of the CT-obstacle in the configuration-time space of $\mathcal{A}$.

b. Adapt the visibility graph method to plan the motion of $\mathcal{A}$ in the configuration-time space of $\mathcal{A}$.

c. Assuming that the search graph is constructed using a sweep-line technique, what would be the time complexity of the method?

2: Show that the planning algorithm of Subsection 1.4.1 still applies if $\mathbf{q}_{goal}$ moves at constant velocity.

3: Modify the velocity tuning method (with polygonal PT-obstacles) presented in Subsection 1.4.3 to handle the dynamic motion planning problem with unspecified arrival time.

4: Consider two convex polygonal objects $\mathcal{A}_1$ and $\mathcal{A}_2$ which translate and rotate freely in $\mathcal{W} = \mathbf{R}^2$. Let $\mathcal{C}$ be their composite configuration space and $\mathcal{CA}_{12}$ be the C-obstacle defined by:

$$\mathcal{CA}_{12} = \{\mathbf{q} \in \mathcal{C} \;/\; \mathcal{A}_1(\mathbf{q}) \cap \mathcal{A}_2(\mathbf{q}) \neq \emptyset\}.$$

Using the approach of Section 2 of Chapter 3, construct a computable expression (in the form of a conjunction of C-constraints) of the predicate

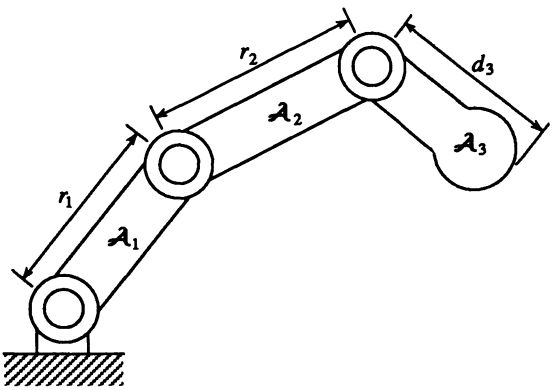


Figure 15. This planar robot arm has three revolute joints (Exercise 9).

$\mathtt{CA}_{12}$ such that:

$$\mathbf{q} \in \mathcal{CA}_{12} \Leftrightarrow \mathtt{CA}_{12}(\mathbf{q}).$$

5: What is the configuration space of a robot that consists of a six-revolute-joint manipulator arm mounted on a platform? (The arm operates in a three-dimensional workspace. The platform translates and rotates freely in a plane.) Discuss the application of various path planning methods to this robot.

6: Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be two polygons that can translate freely at fixed orientation among fixed obstacles. Let τ_1 and τ_2 be two paths defined as in Subsection 2.3.2. Hence, τ_1 (resp. τ_2) is a path of $\mathcal{A}_1$ (resp. $\mathcal{A}_2$) that avoids the fixed obstacles, but ignores the other robot. Assume that both τ_1 and τ_2 are sequences of straight segments. Give a method for constructing the coordination diagram.

7: Write a program based on the techniques described in Subsection 3.3 for computing an approximation of the free space of a robot arm with two revolute joints. Assume that the two links are modeled by straight line segments and that there are no mechanical stops in the joints.

8: Complete the techniques described in Subsection 3.3 to: (1) handle the possible collisions among the various objects composing an artic-

TESLA EXHIBIT 1013, Page 401

ulated robot; (2) to choose the resolution of the decomposition of the EMPTY intervals into small intervals δ_{i,k_i}.

9: Consider a planar manipulator $\mathcal{A} = \{\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3\}$ with three revolute joints (see Figure 15). Its configuration is parameterized by three angles q_1, q_2 and q_3, each varying over $[0, 2\pi)$ (we assume no mechanical stop). The distance between the first two joints (resp. the last two joints) is r_1 (resp. r_2). The maximal distance of any point in $\mathcal{A}_3$ to joint 3 is d_3. Write the formula that computes the largest Euclidean displacement of any point in $\mathcal{A}_3$ when q_1 and q_2 vary over any small intervals of amplitudes δ_1 and δ_2, with q_3 being anywhere in $[0, 2\pi)$.

10: Discuss in general terms a method for solving the problem of planning the coordinated motion of two robot arms carrying the same object (closed-loop kinematic structure) among obstacles.

TESLA EXHIBIT 1013, Page 402

Chapter 10

Dealing with Uncertainty

In the previous chapters we assumed that the robot was able to perfectly control its motions, i.e. to exactly follow the geometrical paths generated by the planner. This assumption is realistic when the workspace is relatively uncluttered and the goal configuration does not have to be achieved too precisely. In those cases, if necessary, uncertainty in robot's control can be taken into account by slightly "growing" the C-obstacles and planning a free path among the grown C-obstacles. Motions that can be planned in this way are often called *gross motions.*

However, when the workspace is too densely occupied by obstacles, growing the C-obstacles may result in a new free space where the initial and goal configurations are no longer connected by a free path. Moreover, even if they are still connected by a path, executing this path may not allow the robot to attain the goal configuration with the desired precision, if the latter is too tight. In those cases, some sensing must be explicitly incorporated into the motion plans in order to both monitor the effects of the robot's motions and command appropriate corrective motions. A motion plan then consists of a combination of motion commands and sensor readings which interact in order to reduce uncertainty

TESLA EXHIBIT 1013, Page 452

and guide the robot toward the goal. Such a plan is usually called a *motion strategy*, and the motions it produces are called *fine motions*.

In general, sensing is not perfect either and does not provide an exact knowledge of the robot's current configuration during execution. Hence, a fundamental difficulty is that uncertainty exists not only at planning time, but also at execution time. It is thus necessary for the planner to anticipate the knowledge that will be available during execution in order to guarantee that this knowledge will be sufficiently precise to make the robot attain the goal region and stop in this region reliably. The motion strategy should not have to be lucky in order to attain the goal recognizably! This typically requires the planner to generate motion commands that will make some specific features of the workspace ("landmarks") reliably identifiable and located by the sensors. Planning a motion strategy may be complicated further by the fact that the geometry of both the robot and the workspace, and hence of the C-obstacle region in the configuration space, is also imprecisely known (uncertainty in model).

In this chapter we investigate the problem of planning the motion of a rigid object among fixed obstacles in the presence of uncertainty. Introducing uncertainty raises new modeling, computational and combinatorial issues which make the problem significantly more difficult than those considered in the previous chapters. This leads us to simplifying the contents of this chapter by making several assumptions presented in Section 1. These assumptions, however, do not denature the fundamental issues underlying motion planning in the presence of uncertainty. In various places, we will discuss the effect of relaxing some of these assumptions.

Section 1 gives a precise formulation of the motion planning problem with uncertainty in control and sensing [1]. It describes force-compliant motion commands which are less sensitive to errors than pure position control (see Subsection 5.3 of Chapter 1). Section 2 introduces a general approach to motion planning with uncertainty, *preimage backchaining*, which was originally proposed in [Lozano-Pérez, Mason and Taylor, 1984]. This approach is described in detail in the rest of the chapter. The concept of preimage — a region of configuration space from which a

[1] Until the last section of this chapter, we will assume that the geometry of the C-obstacle region is exactly known.

motion command is guaranteed to attain a given goal recognizably — is formally defined in Section 3. Simple methods for computing preimages are presented in Section 4. Using these methods as building blocks, one can build an operational planner. To do so, one must select the search space of the planner by discretizing the range of possible directions of motion of the robot. This issue is addressed in Section 5.

The other sections of the chapter present more advanced concepts. Section 6 explores the construction of conditional strategies, i.e. strategies which include conditional branching statements allowing for different motion commands to be executed depending on the values of parameters sensed at execution time. Section 7 investigates ways of improving the ability of a motion command to recognize goal achievement, in order to make it possible constructing larger preimages. In Section 8 we revisit preimage backchaining and we give a broader view of the approach than in Section 2. Finally, Section 9 extends some of the previous results to the case where there is also uncertainty in the input geometric model of the robot and the workspace.

1 Problem Formulation

1.1 Configuration Space

We assume that the robot $\mathcal{A}$ is a polygon that can translate freely at fixed orientation in the plane among polygonal obstacles. Thus, the configuration space is $\mathcal{C} = \mathbf{R}^2$ and the C-obstacle region is polygonal. A configuration is represented as $\mathbf{q} = (x, y)$, where x and y are the coordinates of $\mathcal{A}$'s reference point in the workspace.

We allow $\mathcal{A}$ to touch obstacles[2], so that the space of the robot's motions is the valid space $\mathcal{C}_{valid}$ (see Subsection 9.5, Chapter 2). To avoid pathological cases, we require this space to be a manifold with boundary. This requirement entails that $\mathcal{C}_{contact} = \partial\mathcal{C}_{free}$ and $\mathcal{C}_{valid} = cl(\mathcal{C}_{free})$; hence, a contact between $\mathcal{A}$ and the obstacle region can be broken by an arbitrarily small displacement of $\mathcal{A}$. It also implies that every vertex in the contact space $\mathcal{C}_{contact}$ is the extremity of two edges only; thus, $\mathcal{C}_{contact}$

[2]Mechanical assembly is typical task domain where uncertainty needs to be taken into account. In this domain, contacts between objects are not only part of the goal; they are also helpful to distinguish among possible geometric situations. Most of the research in motion planning with uncertainty has been done in this context.

TESLA EXHIBIT 1013, Page 454

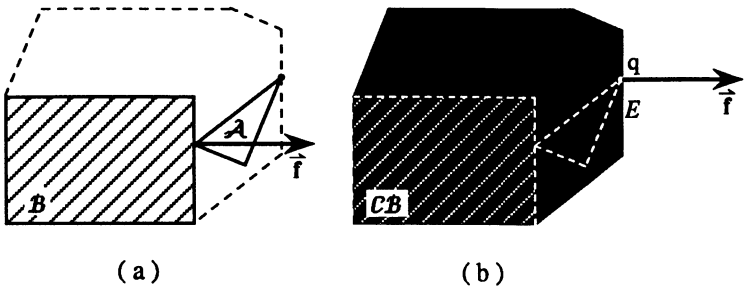


Figure 1. When the moving object $\mathcal{A}$ can only translate in the plane, the workspace $\mathcal{W}$, the configuration space $\mathcal{C}$ and all the tangent spaces of $\mathcal{C}$ are copies of $\mathbf{R}^2$. We equip all these spaces with the Euclidean metric and we represent them as a single copy of $\mathbf{R}^2$. Then a force acting on $\mathcal{A}$, for instance the reaction force exerted by an obstacle, has the same representation in $\mathcal{W}$ and in $\mathcal{C}$, as illustrated in Figures a and b, respectively.

consists of a finite set of disjoint simple polygonal lines. An edge (resp. a vertex) in $\mathcal{C}_{contact}$ is called a **contact edge** (resp. a **contact vertex**). The **outgoing** normal of a contact edge E is the unit vector $\vec{\nu}(E)$ normal to E pointing toward $\mathcal{C}_{free}$. The vector $-\vec{\nu}(E)$ is the **ingoing** normal of E.

Throughout the chapter, n denotes the number of vertices in $\mathcal{C}_{contact}$.

Allowing $\mathcal{A}$ to move in contact space makes it useful to equip the robot with a force sensor for measuring the reaction forces exerted by the obstacles. Force sensing is then a major source of information for the robot to better determine its position relative to the obstacles. It requires, however, that reaction forces be mapped in configuration space. The rest of the subsection addresses this issue.

Under the above assumptions, the workspace $\mathcal{W}$, the configuration space $\mathcal{C}$ and all the tangent spaces of $\mathcal{C}$ are copies of $\mathbf{R}^2$. We equip all these spaces with the Euclidean metric and we use a single copy of $\mathbf{R}^2$ to represent all of them. Then a force acting on $\mathcal{A}$ has the same vector representation in $\mathcal{W}$ and in $\mathcal{C}$ (see Section 10 of Chapter 2). Indeed, a virtual translation of $\mathcal{A}$ by $(\delta x, \delta y)$ in $\mathcal{W}$ induces the same translation of $\mathcal{A}$'s configuration in $\mathcal{C}$. In particular, if $\mathcal{A}$ pushes on an obstacle, the obstacle exerts a reaction force on $\mathcal{A}$. Assuming that $\mathcal{A}$'s configuration

TESLA EXHIBIT 1013, Page 455

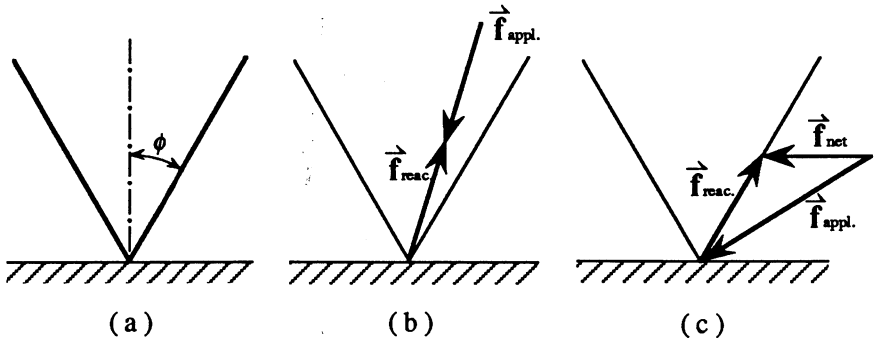


Figure 2. In the presence of friction, the range of possible reaction forces at the contact point between $\mathcal{A}$ and an obstacle consists of all the force vectors applied at the contact point and pointing into a cone, called the friction cone (Figure a). If $\mathcal{A}$ pushes on the obstacle along a line that is contained in the cone, the obstacle produces a reaction force that completely cancels the applied force (Figure b). If $\mathcal{A}$ pushes along a line that is not contained in the cone, the reaction force only partially cancels the applied force and the resulting net force is tangent to the obstacle edge (Figure c).

$\mathbf{q}$ is in the interior of a contact edge E and that the contact between $\mathcal{A}$ and the obstacle is frictionless, the force in configuration space is a vector applied at $\mathbf{q}$ and pointing along the outgoing normal of E, as illustrated in Figure 1.

In practice, contacts are not frictionless. The presence of friction increases the range of reaction forces that the obstacle can produce. Let the contact occur between a vertex of $\mathcal{A}$ and an obstacle edge. The classical Coulomb law of friction represents friction by a cone, called the **friction cone**, which specifies the range of possible reaction forces. The friction cone's axis points along the outgoing normal of the obstacle edge and its sides make an angle $\phi = \tan^{-1}\mu$ $(0 \leq \phi \leq \pi/2)$ with this axis, where μ $(\mu \geq 0)$ is the coefficient of friction between $\mathcal{A}$ and the obstacle at the contact point (see Figure 2.a). Throughout the rest of the chapter, we assume that the static and dynamic friction coefficients have the same value μ and that this value is known and remains fixed for every possible contact point between $\mathcal{A}$ and the obstacles. The range of possible reaction forces at the contact point consists of all the force vectors applied at the contact point and pointing into the friction cone. (The frictionless

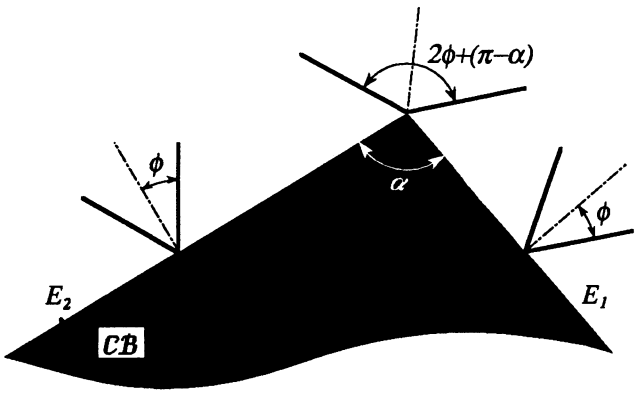


Figure 3. The range of reaction forces at a vertex in contact space can be any non-negative linear combination of the reaction forces that can be generated by the two abutting edges E_1 and E_2. The corresponding friction cone has its sides parallel to the two most extreme rays of the friction cones in E_1 and E_2.

case corresponds to $\phi = 0$.) Hence, if $\mathcal{A}$ pushes on the obstacle along a line that is contained in the cone, the obstacle produces a reaction force that completely cancels[3] the applied force (see Figure 2.b). If, instead, $\mathcal{A}$ pushes along a line that is not contained in the cone, the reaction force only partially cancels the applied force and the resulting net force is tangent to the obstacle edge (see Figure 2.c). The contact between a vertex of $\mathcal{A}$ and an obstacle edge corresponds to a configuration of $\mathcal{A}$ lying in a C-obstacle edge. Since a force has the same representation in $\mathcal{W}$ and $\mathcal{C}$, friction is represented in the C-obstacle edge by the same cone as in the obstacle edge. It is straightforward to extend this result to all the edges of $\mathcal{C}_{contact}$. Since the friction coefficient is constant in the workspace, the friction cone in configuration space has the same angle 2ϕ in every contact edge.

For completeness of the model, we must consider the case when $\mathcal{A}$'s configuration is at a contact vertex X. This case occurs when either a vertex of $\mathcal{A}$ is in contact with an obstacle vertex, or there are two (or more) disconnected contact zones between $\mathcal{A}$ and the obstacles, with different orientations of the friction cones. In the second case, the reaction force at X can be any non-negative linear combination of the reaction forces

[3]We assume that the obstacles are not movable.

that can be generated by the two C-obstacle edges E_1 and E_2 abutting X. In the first case, we assume the same, although the reaction force may not be physically determined (due to the shortcomings of the polygonal representation of the objects). Hence, the friction cone at X is the cone whose sides are respectively parallel to the two most extreme rays of the friction cones in E_1 and E_2 (Figure 3).

The above model of friction can easily be extended to the configuration space of a polyhedral object translating among polyhedral obstacles. But allowing the robot to rotate results in C-obstacles bounded by curved faces. Friction varies in a non-linear fashion over these faces, which makes its description substantially more complicated [Erdmann, 1984].

Throughout this chapter, except in Section 9, we assume that the model of the C-obstacle region is exact.

1.2 Sensing

We assume that the robot $\mathcal{A}$ is equipped with two sensors, a position sensor and a force sensor, which we model below. Other sensors could be modeled in a similar fashion.[4]

The position sensor measures the current configuration of $\mathcal{A}$. The sensed configuration is denoted[5] by $\mathbf{q}^*$, while the current actual configuration is denoted by $\mathbf{q}$. The uncertainty in position sensing is modeled as an open disc $\mathcal{U}_q(\mathbf{q}^*) \subset \mathbf{R}^2$ of fixed radius ρ_q centered at the measured configuration $\mathbf{q}^*$. This means that if the position sensor returns $\mathbf{q}^*$, the current actual configuration $\mathbf{q}$ of $\mathcal{A}$ may be anywhere in $\mathcal{U}_q(\mathbf{q}^*) \cap \mathcal{C}_{valid}$. Notice that although $\mathbf{q}$ can only be in $\mathcal{C}_{valid}$, the sensed configuration $\mathbf{q}^*$ may be in $\mathcal{C} \backslash \mathcal{C}_{valid}$. However, in that case we must have that $\mathcal{U}_q(\mathbf{q}^*) \cap \mathcal{C}_{valid} \neq \emptyset$, since otherwise uncertainty in position sensing would be incorrectly modeled. Since $\mathcal{U}_q$ is a disc, if $\mathbf{q}$ is the actual configuration, then the measured configuration $\mathbf{q}^*$ necessarily lies in $\mathcal{U}_q(\mathbf{q})$.

At this point, our model of uncertainty is worth an explanation. Let the robot be at an actual configuration $\mathbf{q}$ (which of course we do not

[4] Using position and force sensors is appropriate for task domains such as mechanical assembly (see footnote 2). It is less appropriate for mobile robot navigation tasks, which are more likely to involve vision and proximity sensors.

[5] Our convention is to denote a measured quantity by a symbol with the superscript "*" and the corresponding actual quantity by the same symbol without superscript.

TESLA EXHIBIT 1013, Page 458

know) and let the sensed configuration be $\mathbf{q}^*$. The difference $\mathbf{q} - \mathbf{q}^*$ is the sensing *error*, which varies from one measurement to another. Our model of uncertainty bounds the modulus of this error, i.e. $\|\mathbf{q}-\mathbf{q}^*\| < \rho_q$, and accepts all the configurations in $\mathcal{U}_q(\mathbf{q}^*)$ as equally probable ones. Below, we model uncertainty in force sensing and in robot control in a similar fashion. This model is simple, and it is not hard to imagine more involved (and perhaps more realistic) ones. However, it is interesting because it will allow us to neatly define *guaranteed* motion strategies, i.e. strategies which are guaranteed to succeed as long as the errors in control and sensing during execution stay within the bounds specified by the uncertainty regions.

The force sensor measures the force acting on $\mathcal{A}$. The output of the sensor is a vector denoted by $\vec{\mathbf{f}}^*$. The sensor is used to acquire information on whether $\mathcal{A}$ touches obstacles or not and if it touches an obstacle, on the orientation of the outgoing normal of the contact space at the contact configuration. When the sensed force is $\vec{\mathbf{f}}^*$, the actual reaction force acting on $\mathcal{A}$ is a force $\vec{\mathbf{f}} \in \mathcal{U}_f(\vec{\mathbf{f}}^*)$, where $\mathcal{U}_f$ is the force uncertainty region, which is specified as follows. The uncertainty on the sensed force modulus is modeled by $\|\vec{\mathbf{f}}\| \in \Delta_f(\vec{\mathbf{f}}^*)$, where $\Delta_f(\vec{\mathbf{f}}^*)$ is an interval that includes $\|\vec{\mathbf{f}}^*\|$. In addition, the magnitude of the angle between $\vec{\mathbf{f}}$ and $\vec{\mathbf{f}}^*$, if neither is zero, is less than a fixed angle $\varepsilon_f < \pi/2$. Hence, $\vec{\mathbf{f}}$ lies in an open cone of angle $2\varepsilon_f$ whose axis points along $\vec{\mathbf{f}}^*$.

1.3 Robot Control

1.3.1 Motion Command

For a given geometry of $\mathcal{C}_{valid}$, a **motion command** is an application that maps any configuration $\mathbf{q}_s \in \mathcal{C}_{valid}$ (called the **starting configuration** of the motion command) to a nominal motion of the robot. This motion would be the motion of the robot if there was no uncertainty in control.

A motion command $\mathbf{M}$ is described as a pair $(\mathbf{CS}, \mathtt{TC})$. $\mathbf{CS}$ is called the **control statement**. Given a starting configuration $\mathbf{q}_s \in \mathcal{C}_{valid}$, it determines a commanded trajectory τ^c of $\mathcal{A}$, i.e. a curve:

$$\tau^c : t \in [0, +\infty) \mapsto \tau^c(t) \in \mathcal{C}_{valid}$$

with $\tau^c(0) = \mathbf{q}_s$ and t denoting time. TC is called the **termination condition**. The most general form of this condition is:[6]

$$\mathtt{TC} = \mathtt{tp}(t, \mathbf{q}^*[0,t], \vec{\mathbf{f}}^*[0,t])$$

where tp is the **termination predicate**, t denotes the current time, and $\mathbf{q}^*[0,t]$ and $\vec{\mathbf{f}}^*[0,t]$ are the position and force sensing history, respectively, since the beginning of the motion, i.e. :

$$\begin{aligned} \mathbf{q}^*[0,t] &= \{(t', \mathbf{q}^*(t')) \;/\; t' \in [0,t]\} \\ \vec{\mathbf{f}}^*[0,t] &= \{(t', \vec{\mathbf{f}}^*(t')) \;/\; t' \in [0,t]\} \end{aligned}$$

where $\mathbf{q}^*(t')$ and $\vec{\mathbf{f}}^*(t')$ are the sensed configuration and force at time t'. TC is continuously evaluated during the execution of the command and $\mathcal{A}$ instantaneously stops moving when TC becomes `true`. This obviously specifies an ideal behavior which cannot be exactly achieved.

1.3.2 Force-Compliant Control

Throughout the previous chapters, the robot was position-controlled along the planned geometric paths. A drawback of pure position control is that it may be quite sensitive to various kinds of errors. For instance, if the robot is commanded to move along a path close to an obstacle, and if it unexpectedly hits the obstacle (say, because of imperfect control), the normal behavior of a position-controlled robot is to stop. A more interesting behavior would be for the robot to comply with the reaction force exerted by the obstacles (so that it becomes zero), while moving along the projection of the commanded direction in the subspace orthogonal to the reaction force [Mason, 1981]. In the presence of uncertainty, such a compliant behavior is preferable to the previous one, because its ability to attain goals is less sensitive to errors and thus makes it possible to construct motion strategies with fewer motion commands. This behavior requires the robot to be controlled in both position and force.

There exist various models of force-compliant control [Inoue, 1974] [Raibert and Craig, 1981] [Hogan, 1985] [Whitney, 1985]. In order to set up a precise framework for the rest of the chapter, we assume that $\mathcal{A}$'s control follows the so-called **generalized damping** model originally proposed

[6] Less general forms will also be used, for example by not letting tp have access to time and the full sensing history.

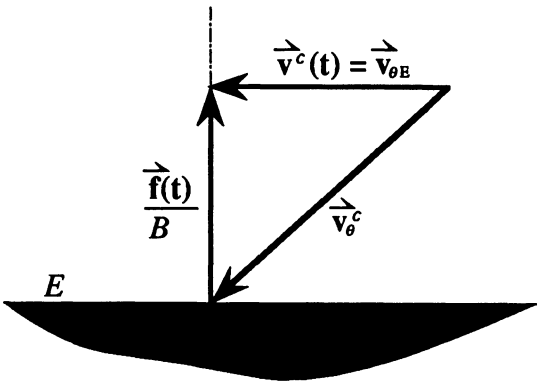


Figure 4. This figure illustrates the robot's nominal behavior in a frictionless contact edge E under the generalized damping control model. The motion is specified by the equation $\vec{f}(t) = B(\vec{v}^c(t) - \vec{v}_\theta^c)$. The robot slides along the edge E at a velocity $\vec{v}_{\theta E}^c$ equal to the projection of $\vec{v}_\theta^c$ on E.

by Whitney [Whitney, 1977]. This means that **CS** is parameterized by a vector $\vec{v}_\theta^c$ of orientation $\theta \in S^1$, called the **commanded velocity**, and that the robot's commanded trajectory τ_θ^c in C is specified by the following first-order differential equation:

$$\vec{f}(t) = B(\vec{v}^c(t) - \vec{v}_\theta^c) \tag{1}$$

where $\vec{f}(t)$ is the actual reaction force exerted by the obstacles on $\mathcal{A}$ at time t, B is a 2×2 matrix called the *damper matrix*, and $\vec{v}^c(t) = d\vec{\tau}_\theta^c(t)/dt$ is the velocity of $\mathcal{A}$ at t. In the following, we take B as a constant diagonal matrix with equal positive diagonal coefficients. We also take $\vec{v}_\theta^c$ as a unit vector, so that it is completely specified by its orientation θ.

In equation (1), the vector $B\vec{v}_\theta^c$ is interpreted as the force exerted by the robot and the vector $B\vec{v}^c(t)$ as the component of this force which is not canceled by the environment and thus produces motion. At every instant t, given the geometry of C_{valid} and the friction angle ϕ, the reaction force $\vec{f}(t)$ is completely determined by the applied force $B\vec{v}_\theta^c$ and the current configuration. Therefore, for a given starting configuration $\mathbf{q}_s$, the generalized damping equation (1) determines a unique trajectory τ_θ^c, which we describe below.

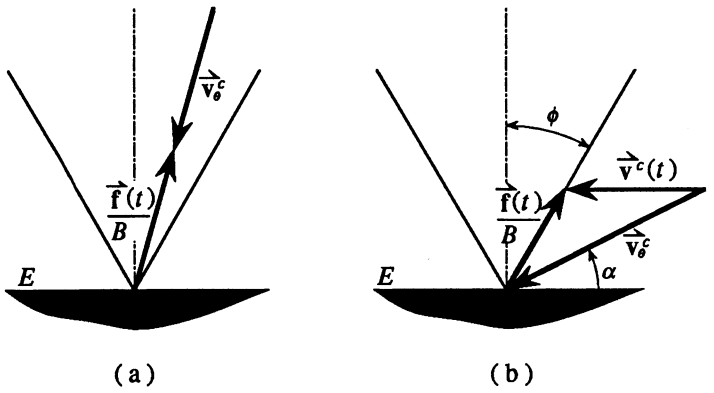


Figure 5. Friction in the contact space increases the range of commanded velocities that lead the robot to stick when it hits a C-obstacle. If the inverted commanded velocity points into the friction cone, the motion sticks to E (Figure a), otherwise it slides along E at velocity $\|\vec{v}^c(t)\| = \|\vec{v}_\theta^c\| \cos\alpha[1 - \tan\phi \tan\alpha]$, where $\alpha < \pi/2$ is the angle between $\vec{v}_\theta^c$ and E (Figure b).

When $\mathcal{A}$ is in free space, it moves at constant velocity $\vec{v}_\theta^c$, since the reaction force is zero. When $\mathcal{A}$ is in contact space in the interior of an edge E, it may move away from E, slide along E, or stick to E. If the projection of $\vec{v}_\theta^c$ on the outgoing normal of E is positive or null, $\mathcal{A}$ moves away or tangentially at velocity $\vec{v}_\theta^c$. Otherwise, let $\vec{v}_{\theta E}^c$ be the projection of $\vec{v}_\theta^c$ on E. If $\vec{v}_{\theta E}^c$ is the null vector (i.e. $\vec{v}_\theta^c$ points perpendicularly to E), then the contact space generates the reaction force $\vec{f}(t) = B\vec{v}_\theta^c$ and $\mathcal{A}$ sticks to E, i.e. $\|\vec{v}^c(t)\| = 0$; otherwise, assuming a frictionless contact space, $\mathcal{A}$ slides along E at velocity $\vec{v}^c(t) = \vec{v}_{\theta E}^c$ (see Figure 4). Since the model of the motion is a first-order one, which ignores inertial forces, it allows the robot velocity to change instantaneously. This idealization is another source of error leading to uncertainty in control.

Friction in the contact space simply results in increasing the range of commanded velocities that cause the robot to stick to a contact edge. With no control error, a generalized damping motion with commanded velocity $\vec{v}_\theta^c$ sticks to an edge E if $-\vec{v}_\theta^c$ lies in the friction cone, i.e. if the magnitude of the angle between $-\vec{v}_\theta^c$ and the outgoing normal $\vec{\nu}(E)$ is less or equal to ϕ. Otherwise, it slides along E. Figure 5 illustrates this behavior.

TESLA EXHIBIT 1013, Page 462

When the robot's configuration reaches a contact vertex X, the motion sticks at X if $-\vec{\mathbf{v}}_\theta^c$ lies in the friction cone at X. If $-\vec{\mathbf{v}}_\theta^c$ does not lie in the friction cone at X, the robot consistently moves away from X (in free or contact space) as if had just reached X by sliding along one of the two adjacent edges.

More formally, let $\vec{\mathbf{f}}_{reac}(\mathbf{q}, \vec{\mathbf{v}}_\theta^c)$ be the function that evaluates to the reaction force exerted by the contact space on the robot at configuration $\mathbf{q}$ when it is commanded along $\vec{\mathbf{v}}_\theta^c$. $\vec{\mathbf{f}}_{reac}(\mathbf{q}, \vec{\mathbf{v}}_\theta^c)$ is defined as follows:

- if $\mathbf{q} \in \mathcal{C}_{free}$, then $\vec{\mathbf{f}}_{reac}(\mathbf{q}, \vec{\mathbf{v}}_\theta^c) = 0$;
- if $\mathbf{q} \in \mathcal{C}_{contact}$, $\mathbf{q}$ is in the interior of an edge E, and $\vec{\mathbf{v}}_\theta^c$ projects positively on the outgoing normal of E, then $\vec{\mathbf{f}}_{reac}(\mathbf{q}, \vec{\mathbf{v}}_\theta^c) = 0$;
- if $\mathbf{q} \in \mathcal{C}_{contact}$, $\mathbf{q}$ is in the interior of an edge E, and $-\vec{\mathbf{v}}_\theta^c$ lies in the friction cone at the contact point, then $\vec{\mathbf{f}}_{reac}(\mathbf{q}, \vec{\mathbf{v}}_\theta^c) = -B\vec{\mathbf{v}}_\theta^c$;
- if $\mathbf{q} \in \mathcal{C}_{contact}$, $\mathbf{q}$ is in the interior of an edge E, and $-\vec{\mathbf{v}}_\theta^c$ projects positively on the outgoing normal of E, but lies outside the friction cone at the contact configuration, then $\vec{\mathbf{f}}_{reac}(\mathbf{q}, \vec{\mathbf{v}}_\theta^c) = \vec{\mathbf{f}}$, with $\vec{\mathbf{f}}$ being the solution of the following system of equations:

$$
\begin{aligned}
&|angle(\vec{\nu}(E), \vec{\mathbf{f}})| = \phi \\
&\vec{\nu}(E) \cdot \vec{\mathbf{f}} = \vec{\nu}(E) \cdot (-B\vec{\mathbf{v}}_\theta^c) \\
&sign(angle(\vec{\nu}(E), \vec{\mathbf{f}})) = sign(angle(\vec{\nu}(E), -B\vec{\mathbf{v}}_\theta^c))
\end{aligned}
$$

- if $\mathbf{q} \in \mathcal{C}_{contact}$, $\mathbf{q}$ is at a contact vertex, and $-\vec{\mathbf{v}}_\theta^c$ lies in the friction cone at this vertex, then $\vec{\mathbf{f}}_{reac}(\mathbf{q}, \vec{\mathbf{v}}_\theta^c) = -B\vec{\mathbf{v}}_\theta^c$;
- if $\mathbf{q} \in \mathcal{C}_{contact}$, $\mathbf{q}$ is at a contact vertex X, and $-\vec{\mathbf{v}}_\theta^c$ does not lie in the friction cone at X, then let $\vec{\mathbf{f}}_1$ and $\vec{\mathbf{f}}_2$ be the reaction forces that would be generated by the two edges abutting X; $\vec{\mathbf{f}}_{reac}(\mathbf{q}, \vec{\mathbf{v}}_\theta^c) = \max(\vec{\mathbf{f}}_1, \vec{\mathbf{f}}_2)$.

(When $\mathbf{q}$ is at a contact vertex, the actual reaction force may not be determined. Then the above definition of $\vec{\mathbf{f}}_{reac}$ is somewhat arbitrary, but it is consistent with our previous definitions.)

At every instant t during a motion commanded along $\vec{\mathbf{v}}_\theta^c$ we have that $\vec{\mathbf{f}}(t) = \vec{\mathbf{f}}_{reac}(\tau_\theta^c(t), \vec{\mathbf{v}}_\theta^c)$. The commanded trajectory τ_θ^c is thus defined by the following differential equation:

$$\frac{d\tau_\theta^c(t)}{dt} = \frac{1}{B}\vec{\mathbf{f}}_{reac}(\tau_\theta^c(t), \vec{\mathbf{v}}_\theta^c) + \vec{\mathbf{v}}_\theta^c$$

with the initial condition $\tau_\theta^c(0) = \mathbf{q}_s$.

The generalized damping model of motion remains applicable in higher-dimensional configuration spaces. However, when the robot is allowed to rotate, this model induces a significantly more complicated behavior.

1.3.3 Uncertainty in Control

Throughout the rest of the chapter, we assume generalized damping control. We denote a motion command by $\mathbf{M} = (\vec{\mathbf{v}}_\theta^c, \mathtt{TC})$, where $\vec{\mathbf{v}}_\theta^c$ is the commanded velocity.

Uncertainty in control is modeled as follows. Let $\vec{\mathbf{v}}_\theta^c$ be the commanded velocity. At any instant t during the motion, the **effective commanded velocity** is a vector $\vec{\mathbf{v}}_\theta(t) \in \mathcal{U}_v(\vec{\mathbf{v}}_\theta^c)$, such that the magnitude of the angle between $\vec{\mathbf{v}}_\theta(t)$ and $\vec{\mathbf{v}}_\theta^c$ is less than a fixed angle $\eta_v < \pi/2$. In other words, $\vec{\mathbf{v}}_\theta(t)$ lies in an open cone of angle $2\eta_v$ whose axis points along $\vec{\mathbf{v}}_\theta^c$. This cone is called the **control uncertainty cone.** In addition, the uncertainty on the velocity modulus is modeled by $\|\vec{\mathbf{v}}_\theta(t)\| \in \Delta_v(\vec{\mathbf{v}}_\theta^c)$, where $\Delta_v(\vec{\mathbf{v}}_\theta^c)$ is an interval[7] (v_1, v_2), with $0 < v_1 < \|\vec{\mathbf{v}}_\theta^c\| = 1 < v_2$. The orientation and the modulus of the effective commanded velocity can vary arbitrarily, but continuously, within these uncertainty bounds.

The motion of the robot is specified by the generalized damping equation (1) with $\vec{\mathbf{v}}_\theta(t)$ substituted for $\vec{\mathbf{v}}_\theta^c$, i.e. $\vec{\mathbf{f}}(t) = B(\vec{\mathbf{v}}(t) - \vec{\mathbf{v}}_\theta(t))$, where $\vec{\mathbf{v}}(t)$ is the actual robot velocity. For a given starting configuration $\mathbf{q}_s$, any continuous map $\vec{\mathbf{v}}_\theta : t \in [0, +\infty) \mapsto \vec{\mathbf{v}}_\theta(t) \in \mathcal{U}_v(\vec{\mathbf{v}}_\theta^c)$ determines a unique actual trajectory $\tau : t \in [0, +\infty) \mapsto \tau(t) \in \mathcal{C}_{valid}$, such that:

$$\frac{d\tau(t)}{dt} = \frac{1}{B}\vec{\mathbf{f}}_{reac}(\tau(t), \vec{\mathbf{v}}_\theta(t)) + \vec{\mathbf{v}}_\theta(t)$$

and $\tau(0) = \mathbf{q}_s$. The only configurations in τ where the actual velocity $\vec{\mathbf{v}}(t) = d\vec{\tau}(t)/dt$ varies discontinuously are those where the trajectory

[7] This interval should not include 0, since according to our model of uncertainty it would allow the robot to stay motionless. However, the lower bound v_1 can be chosen arbitrarily small.

TESLA EXHIBIT 1013, Page 464

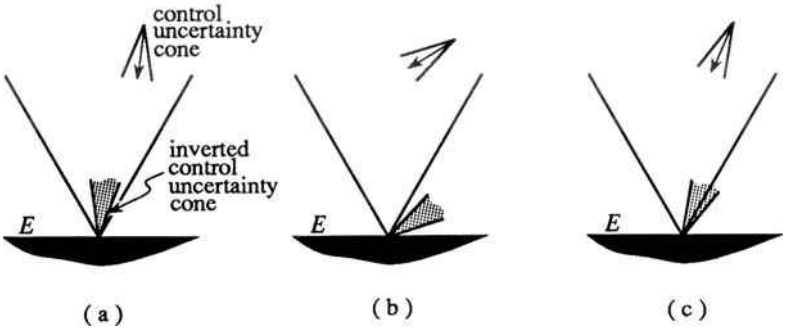


Figure 6. Assume that all the effective commanded velocities lying in the control uncertainty cone project positively on the ingoing normal of a contact edge E. Three cases are possible. If the inverted control uncertainty cone (shown grey) is completely contained in the friction cone (Figure a), the motion is guaranteed to stick along the edge. If the two cones have a null intersection (Figure b), the motion is guaranteed to slide along the edge. If the two cones intersect as in Figure c, the motion may slide or stick.

switches from free to contact space, and conversely. Since there is a finite number of contact edges, τ is piecewise of class C^1.

According to the above model of uncertainty, if $\mathcal{A}$ is in free space, it moves along a trajectory whose tangent at any configuration is contained in the control uncertainty cone anchored at this configuration. If $\mathcal{A}$'s configuration is in a contact edge E, it may move away, slide, or stick, depending on the orientation of the effective commanded velocity $\vec{\mathbf{v}}_\theta$ relative to the outgoing normal of E. In particular, if all vectors lying in the control uncertainty cone project positively on $\vec{\nu}(E)$, then $\mathcal{A}$ is guaranteed to move away from the edge. Instead, if the inverted control uncertainty cone is entirely contained in the friction cone, $\mathcal{A}$ sticks to the edge; if it is completely outside the friction cone and all the vectors in the control uncertainty cone project positively on the ingoing normal of the edge, $\mathcal{A}$ is guaranteed to slide in the direction of the projection of $\vec{\mathbf{v}}_\theta^c$ into E. Figure 6 illustrates the sticking and sliding conditions with uncertainty in control.

We denote the set of all the possible reaction forces that can be generated at configuration $\mathbf{q}$ when the commanded velocity is $\vec{\mathbf{v}}_\theta^c$ by $F_{reac}(\mathbf{q}, \theta)$. We

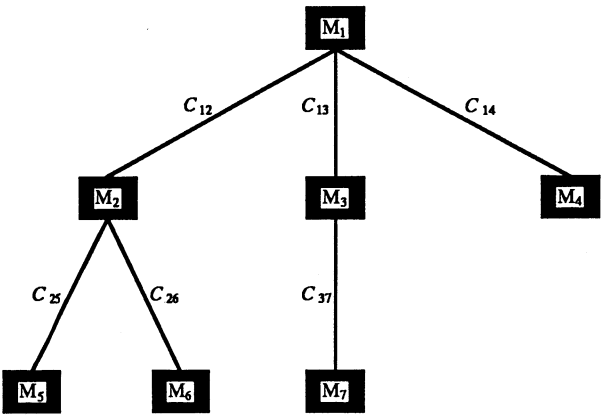


Figure 7. A conditional strategy is described by a tree. The nodes of the tree are labeled by motion commands and its arcs are labeled by conditions. The execution of the strategy consists of sequentially executing the motion commands labeling the nodes of a path (connecting the root to a leaf of the tree) whose arcs are labeled by conditions evaluating to `true`. One possible execution of the strategy depicted in the figure is the following. The motion command $\mathbf{M}_1$ is executed first. Then, if the condition C_{13} evaluates to `true` and C_{12} and C_{14} evaluate to `false`, the motion command $\mathbf{M}_3$ is executed. Since it has a single successor, C_{37} should evaluate to `true` (otherwise the strategy returns failure) and $\mathbf{M}_7$ is executed next.

have that:

$$F_{reac}(\mathbf{q},\theta) \;=\; \bigcup_{\vec{\mathbf{v}}_\theta \in \mathcal{U}_v(\vec{\mathbf{v}}_\theta^c)} \vec{\mathbf{f}}_{reac}(\mathbf{q},\vec{\mathbf{v}}_\theta).$$

A straightforward algorithm computes $F_{reac}(\mathbf{q},\theta)$ in $O(1)$ time.

1.4 Motion Strategies

We describe a **motion strategy** by a tree (Figure 7). The nodes of the tree are motion commands and its arcs are conditions. The execution of the strategy consists of sequentially executing the motion commands labeling the nodes contained in a path connecting the root to a leaf of the tree. After a motion command has been executed, the conditions associated with its outgoing arcs are evaluated and the next command executed is such that the condition of its ingoing arc evaluates to `true`.

TESLA EXHIBIT 1013, Page 466

If several conditions evaluate to `true`, one of the corresponding commands (any one) is executed next. If no condition evaluates to `true`, the execution of the strategy terminates with failure. The execution of the strategy terminates with success after the motion command labeling a leaf of the tree has been executed.

A motion strategy described by a tree containing a single path is called a **sequential strategy**. A motion strategy described by a tree with multiple paths is called a **conditional strategy**. The **number of steps** of a strategy is the maximal number of motion commands that the strategy may have to execute before it can return success.

We assume that the geometry of $\mathcal{C}_{valid}$, the friction angle ϕ, and the uncertainty regions $\mathcal{U}_q$, $\mathcal{U}_f$ and $\mathcal{U}_v$ are given as described above. Given two regions $\mathcal{I}$ and $\mathcal{G}$ of $\mathcal{C}_{valid}$, called the **initial region** and the **goal region**, respectively, the *motion planning problem with uncertainty in control and sensing* is to generate a motion strategy Σ whose execution is guaranteed to return success with $\mathcal{A}$'s configuration actually in $\mathcal{G}$, whenever $\mathcal{A}$'s initial configuration is in $\mathcal{I}$ and the input $\mathcal{C}_{valid}$, ϕ, $\mathcal{U}_q$, $\mathcal{U}_f$, and $\mathcal{U}_v$ are a correct description of the reality. This strategy is said to be **strongly guaranteed**. A complete planner should construct such a strategy whenever one exists and indicate that no such strategy exists otherwise.

Canny and Reif [Canny and Reif, 1987] have shown that the three-dimensional version of the above problem ($\mathcal{C} = \mathbf{R}^3$) is non-deterministic exponential time hard (NEXPTIME-hard) when the number of faces in $\mathcal{C}_{valid}$ is not fixed, which strongly suggests that finding plans can take double exponential time in the worst case[8]. They have also shown that the number of steps in a strategy may grow exponentially with the complexity of $\mathcal{C}_{valid}$. Previously, Natarajan [Natarajan, 1986] had shown that the three-dimensional problem is also PSPACE-hard. On the other hand, no lower-bound time complexity result has been established so far for the problem in two dimensions. There are, however, several upper-bound results obtained with algorithms that we will describe later in this chapter.

[8] Canny and Reif have also shown that in three dimensions, determining whether a given motion command with a given starting configuration is guaranteed to terminate in a goal is NP-hard.

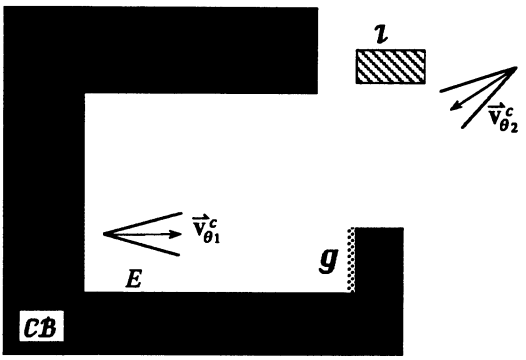


Figure 8. This figure illustrates a sequential two-step motion strategy. The initial region $\mathcal{I}$ is a rectangle in free space and the goal region $\mathcal{G}$ is a contact edge. The first motion is commanded along $\mathbf{v}^c_{\theta_2}$; the second along $\vec{\mathbf{v}}^c_{\theta_1}$.

Until Section 6, we will only consider sequential strategies. Figure 8 illustrates such a strategy. The initial region $\mathcal{I}$ is a rectangle in free space and the goal region $\mathcal{G}$ is an edge in $\mathcal{C}_{contact}$. The strategy consists of two motion commands with commanded velocities $\vec{\mathbf{v}}^c_{\theta_2}$ and $\vec{\mathbf{v}}^c_{\theta_1}$. The command along $\vec{\mathbf{v}}^c_{\theta_2}$ is executed first. (The reverse order on the subscripts results from the backchaining algorithm described in the next section.) With an appropriate termination condition, the motion commanded along $\vec{\mathbf{v}}^c_{\theta_2}$ and executed from within $\mathcal{I}$ is guaranteed to terminate in the horizontal edge denoted by E in the figure; with an appropriate termination condition, the motion commanded along $\vec{\mathbf{v}}^c_{\theta_1}$ and executed from within E is guaranteed to terminate in $\mathcal{G}$. (Note that the conditions labeling the arcs of a strongly guaranteed sequential strategy are irrelevant. They necessarily evaluate to `true`, since otherwise the strategy would not be strongly guaranteed. Hence, we just ignore them.)

Remark: There are problems which admit no strongly guaranteed strategy. This does not mean, however, that the goal $\mathcal{G}$ cannot be achieved from $\mathcal{I}$. Other strategies may achieve $\mathcal{G}$ from $\mathcal{I}$ without guaranteeing success. We might then be interested in generating a **weakly guaranteed** strategy, i.e. a strategy that is guaranteed to either succeed or fail recognizably. This means that the strategy is guaranteed (1) to terminate, (2) to return success only if the goal has actually been achieved, and (3) to return failure only if the goal has not been achieved yet and the

TESLA EXHIBIT 1013, Page 468

remaining non-executed steps in the strategy (if any) have no chance to fortuitously achieve the goal recognizably. Even in cases where strongly guaranteed strategies exist, it may sometimes be preferable to generate weakly guaranteed ones, since the latter are usually simpler (they do not systematically consider the worst cases). The concept of weakly guaranteed strategy has been investigated in [Donald, 1987a and 1988b] under the name of Error Detection and Recovery (EDR) strategy. ∎

2 Preimage Backchaining

In this section we introduce a general approach to motion planning with uncertainty, called *preimage backchaining.* This approach will be investigated in detail in the rest of the chapter. It was originally proposed in [Lozano-Pérez, Mason and Taylor, 1984].

Let $\mathcal{T}$ be a subset of $\mathcal{C}_{valid}$ and $\mathbf{M} = (\vec{\mathbf{v}}_\theta^c, \mathtt{TC})$ be a motion command. A **preimage** of $\mathcal{T}$ for $\mathbf{M}$ is defined as any subset $\mathcal{P}$ of $\mathcal{C}_{valid}$ such that: if $\mathcal{A}$'s configuration is in $\mathcal{P}$ when the execution of $\mathbf{M}$ starts, then it is guaranteed that $\mathcal{A}$ will reach $\mathcal{T}$ and that it will be in $\mathcal{T}$ when TC terminates the motion.

Example: Consider the point-in-hole problem illustrated in Figure 9.a. The region $\mathcal{T}$ is the bottom edge of the depression (the "hole") of the C-obstacle region. Let the commanded velocity be $\vec{\mathbf{v}}_\theta^c$ as shown in the figure. We make the following assumptions: contact space is frictionless; the modulus of the sensed force is accurate; the error in the orientation of the sensed force is bounded by $\varepsilon_f < \pi/4$; and the depth of the hole is greater than $2\rho_q$. The region shown with a striped contour in Figure 9.b is a preimage $\mathcal{P}$ of $\mathcal{T}$ for the generalized damping motion command $\mathbf{M} = (\vec{\mathbf{v}}_\theta^c, \mathtt{TC})$, with TC terminating the motion when the position sensor measures a configuration consistent with $\mathcal{T}$ and the force sensor measures a non-zero force consistent with the outgoing normal of $\mathcal{T}$. More formally, TC is defined as:

$$\mathtt{TC} \equiv [\mathbf{q}^*(t) \in \mathcal{T} \oplus \mathcal{U}_q(\mathbf{0})] \bigwedge [\|\vec{\mathbf{f}}^*(t)\| > 0 \wedge |angle(\vec{\mathbf{f}}^*(t), \vec{\nu}(\mathcal{T}))| < \varepsilon_f]$$

where:

- $\mathbf{q}^*(t)$ and $\vec{\mathbf{f}}^*(t)$ denote the sensed configuration and force at time t, respectively,

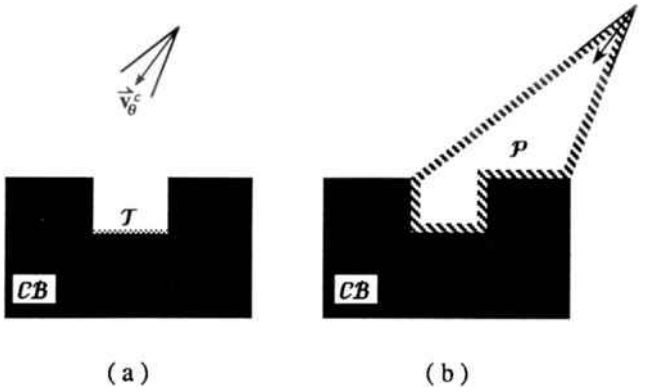


Figure 9. The point-in-hole problem consists of moving the robot's configuration into the edge $\mathcal{T}$ at the bottom of the hole (Figure a). The region shown in striped contour in Figure b depicts a preimage of $\mathcal{T}$ for the generalized damping motion command $(\vec{v}_\theta^c, \mathtt{TC})$. The commanded velocity $\vec{v}_\theta^c$ and the control uncertainty cone are shown in the figure. The termination condition TC is defined in the text.

- $\mathcal{T} \oplus \mathcal{U}_q(\mathbf{0})$ is the edge $\mathcal{T}$ "grown" by ρ_q. (The symbol $\oplus$ denotes the Minkowski operator for affine set addition.)

The execution of M generates a trajectory that (in free space) is locally contained in the control uncertainty cone and (in contact space) slides over edges that are not orthogonal to a direction contained in the control uncertainty cone. Thus, it is guaranteed to reach $\mathcal{T}$ whenever the starting configuration of $\mathcal{A}$ is in $\mathcal{P}$. Since $\varepsilon_f < \pi/4$, the second condition of TC, $\|\vec{f}^*(t)\| > 0 \wedge |angle(\vec{f}^*(t), \vec{\nu}(\mathcal{T}))| < \varepsilon_f$, guarantees that the motion will terminate in one of the three horizontal contact edges. Since the depth of the hole is greater than $2\rho_q$, the first condition, $\mathbf{q}^*(t) \in \mathcal{T} \oplus \mathcal{U}_q(\mathbf{0})$, allows TC to distinguish between the bottom edge $\mathcal{T}$ and the two horizontal side edges bordering the entrance of the hole. Hence, TC will stop the motion in $\mathcal{T}$, and $\mathcal{P}$ is a preimage of $\mathcal{T}$ for M. ∎

Let PREIMAGE be an algorithm that, given a commanded velocity $\vec{v}_\theta^c$ and a region $\mathcal{T} \subset \mathcal{C}_{valid}$, returns a region $\mathcal{P}$ and a termination condition TC such that $\mathcal{P}$ is a preimage of $\mathcal{T}$ for the motion command $(\vec{v}_\theta^c, \mathtt{TC})$. Given the initial and goal regions, $\mathcal{I}$ and $\mathcal{G}$, **preimage backchaining** essentially proceeds according to the following algorithm:

TESLA EXHIBIT 1013, Page 470

```
procedure PB;
    begin
        P_0 ← G; i ← 0;
        while I ⊈ P_i do
            begin
                i ← i + 1;
                select θ_i in S^1;
                P_i ← PREIMAGE(v^c_{θ_i}, P_{i-1});
            end;
    end;
```

Since it selects a commanded velocity at each iteration, the backchaining process in fact consists of searching an infinite tree of preimages. If the search terminates successfully, its outcome is a sequence of preimages $\mathcal{P}_1, \mathcal{P}_2, ..., \mathcal{P}_r$ such that:

- $\mathcal{P}_i$, $\forall i \in [1, r]$, is the preimage of $\mathcal{P}_{i-1}$ (with $\mathcal{P}_0 = \mathcal{G}$) computed by PREIMAGE for the selected commanded velocity $\vec{\mathbf{v}}^c_{\theta_i}$ and a termination condition $\mathtt{TC}_i$ returned by PREIMAGE;
- $\mathcal{I} \subseteq \mathcal{P}_r$.

The reverse sequence of the motion commands, $(\mathbf{M}_r, \mathbf{M}_{r-1}, ..., \mathbf{M}_1)$, with $\mathbf{M}_i = (\vec{\mathbf{v}}^c_{\theta_i}, \mathtt{TC}_i)$, is the generated (sequential, strongly guaranteed) r-step motion strategy.

Example: Figure 10 illustrates an idealized preimage backchaining process with the example previously given in Figure 8. First, a commanded velocity $\vec{\mathbf{v}}^c_{\theta_1}$ is selected and the preimage $\mathcal{P}_1$ of $\mathcal{G}$ is computed (Figure 10.a). This preimage does not contain $\mathcal{I}$ and is taken as an intermediate goal. Second, a commanded velocity $\vec{\mathbf{v}}^c_{\theta_2}$ is selected and the preimage $\mathcal{P}_2$ of $\mathcal{P}_1$ is computed (Figure 10.b). $\mathcal{P}_1$ is so thin at one end that the termination condition of this motion command can reliably recognize $\mathcal{P}_1$'s achievement only by detecting contact (using force sensing) in the edge denoted by E, which is part of $\mathcal{P}_1$. Since $\mathcal{P}_2$ contains $\mathcal{I}$, the planning problem is solved with a sequence of two motions. Interestingly, the preimage backchaining process has resulted in identifying and using E as an intermediate "landmark" to help the reliable progression of the robot toward $\mathcal{G}$. Of course, in practice, the backchaining process would probably not generate such a strategy directly; rather, it would try several commanded velocities, some unsuccessfully. ∎

TESLA EXHIBIT 1013, Page 471

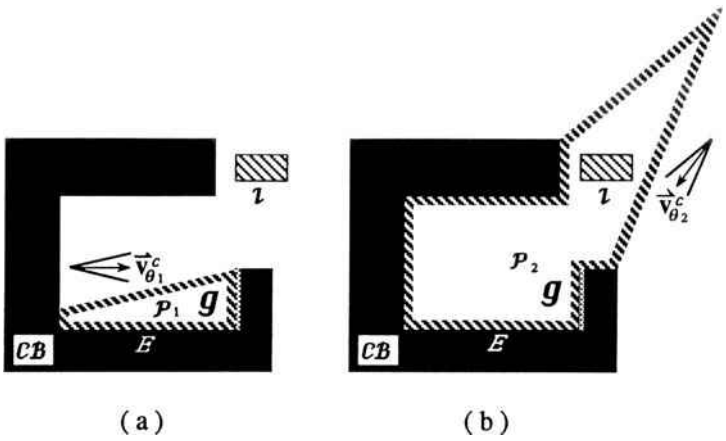


Figure 10. This figure illustrates an ideal preimage backchaining process with the example of Figure 8. First, the commanded velocity $\vec{v}^c_{\theta_1}$ is selected and the preimage $\mathcal{P}_1$ of $\mathcal{G}$ is computed. This preimage does not contain $\mathcal{I}$. It is taken as an intermediate goal. The commanded velocity $\vec{v}^c_{\theta_2}$ is selected and the preimage $\mathcal{P}_2$ of $\mathcal{P}_1$ is computed. $\mathcal{P}_2$ contains $\mathcal{I}$ and the planning problem is solved.

The preimage backchaining algorithm PB given above is obviously oversimplified. In order to construct an effective planner, the following three issues must be investigated:

- *Computation of maximal preimages:* For a given commanded velocity and a given region $\mathcal{T}$, the ideal algorithm PREIMAGE would compute the maximal preimage of $\mathcal{T}$ over all possible termination conditions and return a termination condition able to recognize the achievement of the goal. Indeed, a large preimage has more chance to include the initial region $\mathcal{I}$ than a smaller preimage; in addition, if it is considered recursively as an intermediate goal, a large preimage has more chance to admit large preimages than a smaller one. Thus, considering maximal preimages may reduce the size of the search space; it may also produce shorter strategies. On the other hand, using non-maximal preimages may not guarantee that a motion strategy will be ultimately generated, even if one exists. But do maximal preimages exist?
- *Choice of the commanded velocities:* The problem of generating a sequential strategy can be transformed into the combinatorial problem of

searching a tree by selecting commanded velocities from a finite set of values in S^1. The root of this tree is the goal region $\mathcal{G}$, and every other node is a preimage region. Each arc is a motion command, connecting a region to its preimage computed by PREIMAGE for this command. However, a predefined discretization of S^1 may not guarantee the planner to generate a sequential strategy whenever one exists. Is there a way to discretize S^1 in a such a way that we are sure not to miss any important velocity orientation?

- *Generation of conditional strategies:* Searching a tree of preimages, as suggested above only permits the planner to generate sequential strategies. There exist problems whose only solutions are conditional strategies (see Section 6). A conditional branching occurs when a motion is guaranteed to achieve one of several intermediate goals, without knowing which one in advance, in such a way that it is guaranteed at planning time that it will be possible to recognize at execution time which one has been achieved. For these problems, another search space must be considered, one which allows the planner to compute the preimages of sets of goals, rather than of individual goals. How should this space be defined and searched?

These issues are addressed in the next sections (Sections 3 through 8).

Note that the preimage backchaining approach is applicable even when there is no uncertainty. It also applies with control models other than generalized damping. Only the computation of the preimages (which we have not described so far) and the parameters to discretize in order to build the search space would be changed. In fact, preimage backchaining is based on a very general planning idea. A preimage can be regarded as a precondition (ideally, the weakest one) whose satisfaction before executing an action guarantees that some postcondition will be satisfied after the action is executed. The same idea has been used previously as a tool to analyze algorithms [Hoare, 1969] and as the basis of the "goal regression" planning method in Artificial Intelligence [Waldinger, 1977] [Nilsson, 1980].

In the following, the word "goal" designates either the goal region $\mathcal{G}$ of the motion planning problem, or any preimage taken as an intermediate goal. A goal is generically denoted by $\mathcal{T}$.

3 Formal Definition of a Preimage

The notion of a preimage of a goal $\mathcal{T}$ for a motion command $\mathbf{M} = (\vec{\mathbf{v}}_\theta^c, \mathtt{TC})$ combines two basic concepts, known as **goal reachability** and **goal recognizability** [Erdmann, 1984].

Goal reachability relates to the fact that any trajectory obtained by executing a motion commanded along $\vec{\mathbf{v}}_\theta^c$ from a preimage of $\mathcal{T}$ should reach $\mathcal{T}$. Given a starting configuration, $\vec{\mathbf{v}}_\theta^c$ only determines a **commanded trajectory** τ_θ^c. Due to uncertainty in control, any execution produces an **actual trajectory** τ that is slightly different. The planner must be certain that all the possible actual trajectories consistent with both $\vec{\mathbf{v}}_\theta^c$ and $\mathcal{U}_v$ will reach $\mathcal{T}$ at some instant.

Goal reachability, however, is not enough, since the termination condition could just let the robot traverse the goal without stopping it or, instead, it could halt the motion prematurely before the goal has been reached. The planner must also be certain that the termination condition TC will stop $\mathcal{A}$ in $\mathcal{T}$. This is the goal recognizability issue. TC is an observer of the actual trajectory τ being executed. Since sensing is imperfect, it perceives the actual trajectory as an **observed trajectory** τ^* which is most likely to be neither the commanded one, nor the actual one. Thus, the problem for the planner is to be sure that, for every possible observed trajectory τ^*, (1) TC becomes `true` at some instant t_f, and (2) when TC first becomes `true`, all the actual trajectories τ consistent with τ^* (i.e. the trajectories which may be observed as τ^* given the uncertainty in sensing) have reached the goal, i.e. $\tau(t_f) \in \mathcal{T}$.

This discussion leads us to define the notion of trajectory directory:

DEFINITION 1: *The* **directory of actual trajectories** *for a region $\mathcal{P}$ and a commanded velocity $\vec{\mathbf{v}}_\theta^c$ is the set $\mathcal{D}_\theta(\mathcal{P})$ of all the trajectories starting inside $\mathcal{P}$ and consistent with $\vec{\mathbf{v}}_\theta^c$, the control uncertainty $\mathcal{U}_c$, and the C-obstacle region. More formally, $\mathcal{D}_\theta(\mathcal{P})$ is the set of all the trajectories $\tau : t \in [0, +\infty) \mapsto \tau(t) \in \mathcal{C}_{valid}$ piecewise of class C^1, such that $\tau(0) \in \mathcal{P}$ and (see Subsection 1.3.2):*

$$\frac{d\vec{\tau}(t)}{dt} = \frac{1}{B}\vec{\mathbf{f}}_{reac}(\tau(t), \vec{\mathbf{v}}_\theta(t)) + \vec{\mathbf{v}}_\theta(t)$$

with $\vec{\mathbf{v}}_\theta(t) \in \mathcal{U}_v(\vec{\mathbf{v}}_\theta^c)$ being of class C^0.

We have that:

- $\mathcal{D}_\theta(\mathcal{P}) = \bigcup_{\mathbf{q}_s \in \mathcal{P}} \mathcal{D}_\theta(\{\mathbf{q}_s\})$,
- $\mathcal{D}_\theta(\mathcal{P}_1 \cup \mathcal{P}_2) = \mathcal{D}_\theta(\mathcal{P}_1) \cup \mathcal{D}_\theta(\mathcal{P}_2)$.

With each trajectory τ in $\mathcal{D}_\theta(\mathcal{P})$ we associate two functions, $\vec{\mathbf{v}}_\tau(t)$ and $\vec{\mathbf{f}}_\tau(t)$. They respectively evaluate at each instant t to the effective commanded velocity that produces τ and to the reaction force exerted on the robot. We have that: $\vec{\mathbf{f}}_\tau(t) = \vec{\mathbf{f}}_{reac}(\tau(t), \vec{\mathbf{v}}_\tau(t))$.

We represent an observed trajectory τ^* by a pair $(\mathbf{q}_{\tau^*}(t), \vec{\mathbf{f}}_{\tau^*}(t))$, where $\mathbf{q}_{\tau^*}$ and $\vec{\mathbf{f}}_{\tau^*}$ are functions mapping the elapsed time t since the beginning of the motion to the sensed configuration and the sensed force at this instant.

The consistency between an actual and an observed trajectory is defined as follows:

DEFINITION 2: *An actual trajectory τ and an observed trajectory $\tau^* = (\mathbf{q}_{\tau^*}, \vec{\mathbf{f}}_{\tau^*})$ are* **consistent** *if and only if:*

$$\forall t \in [0, +\infty) : (\tau(t), \vec{\mathbf{f}}_\tau(t)) \in \mathcal{U}_q(\mathbf{q}_{\tau^*}(t)) \times \mathcal{U}_f(\vec{\mathbf{f}}_{\tau^*}(t)).$$

$\mathcal{K}^*_{traj}(\tau) \stackrel{\text{def}}{=} \{\tau^* \ / \ \tau^* \text{ and } \tau \text{ are consistent}\}$ *is the set of all the possible observations of τ.*

The termination condition $\mathtt{TC} = \mathtt{tp}(t, \mathbf{q}^*[0,t], \vec{\mathbf{f}}^*[0,t])$ applies to the observed trajectory. The directory $\mathcal{D}_\theta(\mathcal{P})$ contains all the possible actual trajectories. Each trajectory $\tau \in \mathcal{D}_\theta(\mathcal{P})$ may be observed by TC as any trajectory in $\mathcal{K}^*_{traj}(\tau)$. We define $\mathtt{Achieve}(\mathcal{T}, \mathrm{M}, \mathcal{P})$ as the condition expressing that the execution of M is guaranteed to terminate in $\mathcal{T}$ whenever the starting configuration is in $\mathcal{P}$:

DEFINITION 3: *Let* $\mathrm{M} = (\vec{\mathbf{v}}^c_\theta, \mathtt{tp}(t, \mathbf{q}^*[0,t], \vec{\mathbf{f}}^*[0,t]))$.

$$\begin{aligned}\mathtt{Achieve}(\mathcal{T}, \mathrm{M}, \mathcal{P}) \stackrel{\text{def}}{=}\ & \forall \tau \in \mathcal{D}_\theta(\mathcal{P}), \forall \tau^* \in \mathcal{K}^*_{traj}(\tau) : \\ & . \exists t \text{ verifying } \mathtt{tp}(t, \mathbf{q}_{\tau^*}[0,t], \vec{\mathbf{f}}_{\tau^*}[0,t]); \\ & . \tau(t_f) \in \mathcal{T}, \text{ where :} \\ & \qquad t_f = \inf\{t \ / \ \mathtt{tp}(t, \mathbf{q}_{\tau^*}[0,t], \vec{\mathbf{f}}_{\tau^*}[0,t])\}.\end{aligned}$$

This yields the following definition of a preimage:

DEFINITION 4: *A* **preimage** *of a goal* $\mathcal{T}$ *for the motion command* M *is any region* $\mathcal{P}$ *verifying* $\texttt{Achieve}(\mathcal{T}, \mathrm{M}, \mathcal{P})$.

The following properties are direct consequences of this definition:

- If $\mathcal{P}$ is a preimage of $\mathcal{T}$ for M, then any subset of $\mathcal{P}$ is also a preimage of $\mathcal{T}$ for M.
- If $\mathcal{P}_1$ and $\mathcal{P}_2$ are both preimages of $\mathcal{T}$ for M, then $\mathcal{P}_1 \cup \mathcal{P}_2$ is also a preimage of $\mathcal{T}$ for M.

These two properties directly yield the notion of maximal preimage of a goal for a given motion command:[9]

DEFINITION 5: *The region*

$$\mathcal{P}_{\mathrm{M}}^{max}(\mathcal{T}) = \{\mathbf{q}_s \;/\; \texttt{Achieve}(\mathcal{T}, \mathrm{M}, \{\mathbf{q}_s\})\}$$

is the **maximal preimage** *of* $\mathcal{T}$ *for* M.

In general:

$$\mathcal{P}_{\mathrm{M}}^{max}(\mathcal{T}_1) \cup \mathcal{P}_{M}^{max}(\mathcal{T}_2) \subseteq \mathcal{P}_{\mathrm{M}}^{max}(\mathcal{T}_1 \cup \mathcal{T}_2).$$

Indeed, there may be configurations from which a motion command is guaranteed to reach and terminate in one of the two regions $\mathcal{T}_1$ and $\mathcal{T}_2$, without knowing which one in advance.

Given a commanded velocity $\vec{\mathbf{v}}_\theta^c$, the size of the maximal preimage of a goal $\mathcal{T}$ depends on the ability of the termination condition to recognize the achievement of $\mathcal{T}$. We will analyze this ability for several termination conditions in Section 7. Before that, we describe two practical methods which compute preimages for relatively simple termination conditions and we explore the construction of a motion planner using these two methods.

[9] This notion of maximal preimage is defined for a *fixed* motion command, hence a fixed termination condition. We will see that the maximal preimage for a goal $\mathcal{T}$ may not exist if we allow the termination condition to *vary*.

TESLA EXHIBIT 1013, Page 476

4 Preimage Computation

4.1 Separation of Reachability and Recognizability

As suggested in Section 2, for a given commanded velocity $\vec{\mathbf{v}}_\theta^c$ and a given goal region $\mathcal{T}$, the ideal PREIMAGE algorithm would compute the maximal preimage of $\mathcal{T}$ over all the possible termination conditions and return the corresponding termination condition. However, we will see later that: (1) if we allow the termination condition to vary, there does not always exist a maximal preimage; (2) if there exists one, it may not be unique; and (3) if there exists a unique maximal preimage, it may depend in a subtle fashion on the knowledge used by the termination condition. Most of the difficulties related to the computation of "large" preimages are due to the interdependence between goal reachability and goal recognizability, i.e. to the fact that deciding at each instant whether the robot is in the goal, or not, may depend on past history.

Erdmann [Erdmann, 1984] suggested approaching the preimage computation problem by restricting the recognition power of the termination condition so that the reachability and recognizability issues can be treated separately. The basic idea is to construct a termination condition which (1) may evaluate to `true` only if the robot is in the goal $\mathcal{T}$ and (2) is guaranteed to evaluate to `true` when a certain subset $\mathcal{S} \subseteq \mathcal{T}$ is attained. Then it remains to compute a region $\mathcal{L}$ from where the robot is guaranteed to reach $\mathcal{S}$ when commanded along some given velocity $\vec{\mathbf{v}}_\theta^c$. $\mathcal{L}$ is called a *backprojection* of $\mathcal{S}$ for $\vec{\mathbf{v}}_\theta^c$ (see Subsections 4.2 and 4.3). It is a preimage of $\mathcal{T}$.

In Subsections 4.4 and 4.5 we describe two methods based on this general idea. Both methods are easily implementable and can serve as building blocks for simple motion planners. However, since they restrict the recognition power of the termination conditions, such planners are not complete.

4.2 Notion of Backprojection

A backprojection is defined as follows [Erdmann, 1984]:

DEFINITION 6: *Let:*

$$\texttt{Reach}(\mathcal{S}, \theta, \mathcal{L}) \stackrel{\text{def}}{=} \forall \tau \in \mathcal{D}_\theta(\mathcal{L}), \exists t \geq 0 : \tau(t) \in \mathcal{S}.$$

A **backprojection** *of* $\mathcal{S}$ *for* $\vec{\mathbf{v}}_\theta^c$ *is any region* $\mathcal{L}$ *verifying* $\mathtt{Reach}(\mathcal{S}, \theta, \mathcal{L})$.

If two regions $\mathcal{L}_1$ and $\mathcal{L}_2$ are backprojections of $\mathcal{S}$ for $\vec{\mathbf{v}}_\theta^c$, then their union $\mathcal{L}_1 \cup \mathcal{L}_2$ is also a backprojection of $\mathcal{S}$ for $\vec{\mathbf{v}}_\theta^c$. Hence:

DEFINITION 7: *The region* $\{\mathbf{q}_s \in \mathcal{C}_{valid} \;/\; \mathtt{Reach}(\mathcal{S}, \theta, \{\mathbf{q}_s\})\}$ *is the* **maximal backprojection** *of* $\mathcal{S}$ *for* $\vec{\mathbf{v}}_\theta^c$. *It is denoted by* $\mathcal{L}_\theta(\mathcal{S})$.

From now on, we will only consider backprojections that are maximal. Hence, we will usually omit this qualifier.

A backprojection of a goal differs from a preimage by the fact that it ignores the need to recognize the achievement of the goal. Clearly, any preimage of a goal $\mathcal{T}$ is included in the backprojection of $\mathcal{T}$ for the same commanded velocity. Hence, the backprojection of $\mathcal{T}$ for any given commanded velocity is an upper bound of the maximal preimage, if any, of the same goal for the same commanded velocity, over all possible termination conditions.

As with maximal preimages, in general:

$$\mathcal{L}_\theta(\mathcal{S}_1) \cup \mathcal{L}_\theta(\mathcal{S}_2) \subseteq \mathcal{L}_\theta(\mathcal{S}_1 \cup \mathcal{S}_2).$$

Indeed, there may be configurations from which a motion commanded along $\vec{\mathbf{v}}_\theta^c$ is guaranteed to reach one of the two regions $\mathcal{S}_1$ and $\mathcal{S}_2$, without knowing which one in advance.

Figure 11 shows the backprojections (striped contours) for a single contact edge $\mathcal{S}$ for two commanded velocities $\vec{\mathbf{v}}_{\theta_1}^c$ and $\vec{\mathbf{v}}_{\theta_2}^c$.

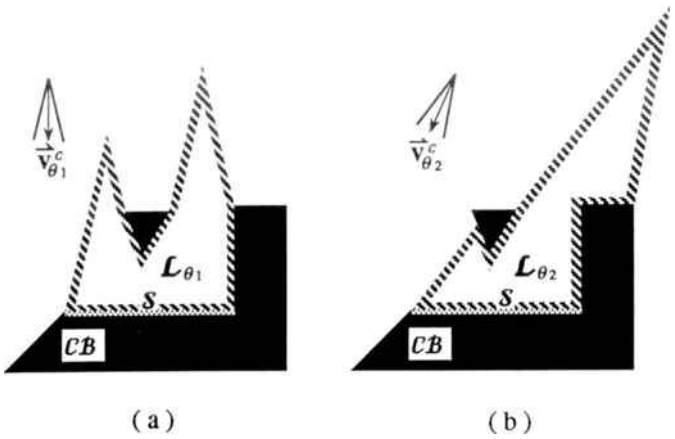
4.3 Computation of a Backprojection

Given a commanded velocity $\vec{\mathbf{v}}_\theta^c$, the following algorithm computes the maximal backprojection of a goal region[10] $\mathcal{S}$ consisting of a single contact edge[11] [Erdmann, 1984]:

[10] Throughout this subsection, we call the region $\mathcal{S}$ a goal, although it is more likely to be a subset of a goal.

[11] If $\mathcal{S}$ is a portion of a contact edge, the algorithm applies by treating the two endpoints of $\mathcal{S}$ as additional vertices of $\mathcal{C}_{contact}$.

TESLA EXHIBIT 1013, Page 478



(a) (b)

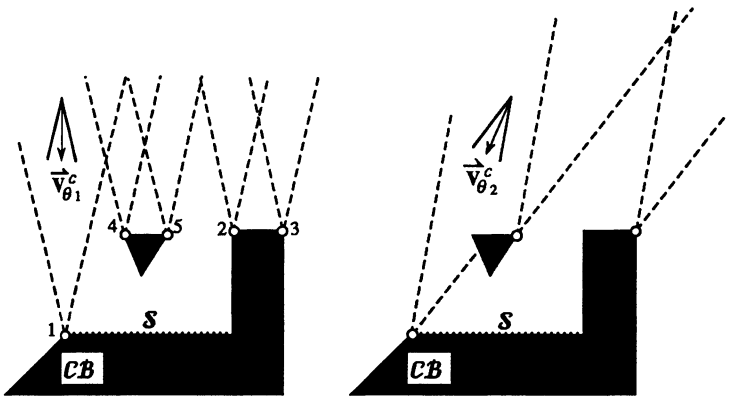
Figure 11. The regions outlined in striped contour in both figures are the maximal backprojections $\mathcal{L}_{\theta_1}(\mathcal{S})$ and $\mathcal{L}_{\theta_2}(\mathcal{S})$ of the edge $\mathcal{S}$ for the commanded velocities $\vec{v}^c_{\theta_1}$ and $\vec{v}^c_{\theta_2}$, respectively. Any motion commanded along $\vec{v}^c_{\theta_1}$ (resp. $\vec{v}^c_{\theta_2}$) and starting from within $\mathcal{L}_{\theta_1}(\mathcal{S})$ (resp. $\mathcal{L}_{\theta_2}(\mathcal{S})$) is guaranteed to reach $\mathcal{S}$.

1. Consider every vertex in $\mathcal{C}_{contact}$. Mark every non goal vertex at which sticking is possible (i.e. such that the friction cone intersects the inverted control uncertainty cone). Mark every goal vertex if sticking is possible in the non goal edge abutting this vertex. Mark every goal vertex if sliding away from the vertex in the non goal edge is possible.

2. At every marked vertex erect two rays parallel to the sides of the inverted control uncertainty cone. Compute the intersection of these rays among themselves and with $\mathcal{C}_{contact}$. Ignore each ray beyond the first intersection. The remaining segment in each ray is called a **free edge**. The marked vertex at which the ray was erected is called the **anchor vertex** of the free edge.

3. Trace out the backprojection region. To that purpose, trace the edge $\mathcal{S}$ in the direction that leaves the C-obstacle region on the right-hand side, until an endpoint of $\mathcal{S}$ is attained; then, let all the contact and free edges incident at this point be sorted clockwise in a circular list; trace the first edge following $\mathcal{S}$ in this list; proceed in the same way until $\mathcal{S}$ is encountered again.

TESLA EXHIBIT 1013, Page 479



(a) (b)

Figure 12. This figure illustrates the computation of the two backprojections shown in Figure 11, by Erdmann's algorithm. Figure a shows both the vertices which are marked at step 1 for the commanded velocity $\vec{v}^c_{\theta_1}$ (small circles), and the rays that are erected at step 2 from these points. Vertex 1 is marked because it is possible to slide away from $\mathcal{S}$. Vertices 2 through 5 are marked because they are non goal vertices where sticking is possible. Figure b shows the marked vertices and the erected rays for the commanded velocity $\vec{v}^c_{\theta_2}$.

This algorithm requires the backprojection to be bounded, which can be achieved by imposing that $\mathcal{C}_{valid}$ or (if $\eta_v \neq 0$) the C-obstacle region be bounded. In addition, the algorithm does not cover some particular cases, which we will discuss below.

The construction of the backprojections shown in Figure 11 by this algorithm is illustrated in Figure 12. The vertices marked at step 1 are depicted as small circles. Erecting the inverted control uncertainty cone at the marked vertices produces constraints — the free edges — which prevent the backprojection from containing configurations from where a motion may reach a vertex or an edge in which it can stick or slide away from the goal with no possibility of returning to it. Notice that the algorithm only constructs the boundary of the backprojection of $\mathcal{S}$. The region lying within this boundary and the portions of contact edges contained in this boundary are part of the backprojection. Assuming that the control uncertainty cone is open, each free edge in the boundary is

TESLA EXHIBIT 1013, Page 480

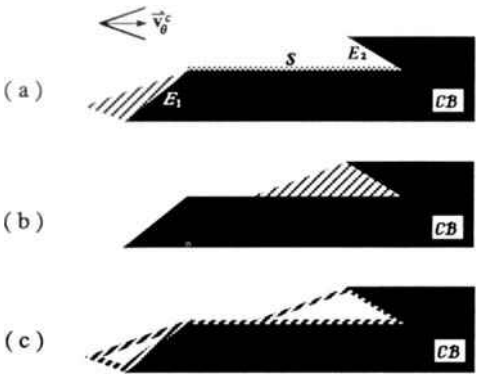


Figure 13. This figure illustrates special cases which may occur when some of the directions contained in the control uncertainty cone project negatively on the ingoing normal of the goal edge $\mathcal{S}$.

also part of the backprojection, with the exception of its anchor vertex if the latter is a non goal vertex.

The above algorithm treats only the case where all the directions contained in the control uncertainty cone project positively on the ingoing normal of $\mathcal{S}$. In the case where some (or all) directions project negatively, we could make the algorithm simply return $\mathcal{S}$ as the backprojection. This is correct, since $\mathcal{S}$ is always a (possibly non-maximal) backprojection of itself, but it is too conservative, as we illustrate in Figure 13.

The striped triangle in Figure 13.a contains starting configurations from which the motion along $\vec{\mathbf{v}}_\theta^c$ is guaranteed to reach the left vertex of $\mathcal{S}$ by sliding along the edge E_1 abutting this vertex (assuming frictionless contact space for simplicity) and therefore is part of $\mathcal{L}_\theta(\mathcal{S})$. The striped triangle in Figure 13.b is also part of $\mathcal{L}_\theta(\mathcal{S})$, since all the motions starting in it ultimately reach the right vertex of $\mathcal{S}$ eventually by sliding along E_2. Figure 13.c shows $\mathcal{L}_\theta(\mathcal{S})$. It contains the two previous triangles and the edge $\mathcal{S}$. (Notice that each of the two triangles is a backprojection of one of the two endpoints of $\mathcal{S}$. Removing these two vertices from the goal, i.e. considering the open edge, rather than the closed one, as the goal, would result in $\mathcal{S}$ being its own backprojection.) Erdmann [Erdmann, 1986] described a modification of his original algorithm for handling the cases illustrated in Figure 13 in a general way.

TESLA EXHIBIT 1013, Page 481

Erdmann's algorithm has been generalized by Canny and Donald [Donald, 1987a] to an algorithm that generates the maximal backprojection of any region $\mathcal{S}$ described as the union of contact edges and polygonal regions in $\mathcal{C}_{valid}$. The principle of the algorithm is the following. It sweeps a line L across the plane perpendicularly to the commanded velocity $\vec{\mathbf{v}}_\theta^c$. The sweep starts at a position of L where it is tangent to $\mathcal{S}$, with $\mathcal{S}$ lying entirely on the side of L pointed to by the vector $-\vec{\mathbf{v}}_\theta^c$. The sweep proceeds in the direction of the vector $-\vec{\mathbf{v}}_\theta^c$. During the sweep, the algorithm maintains the status of L, i.e. the description of its intersection with $\mathcal{C}_{contact}$, $\mathcal{S}$ and free edges erected from contact vertices and goal vertices. This status changes qualitatively at events occurring when L passes through a vertex of $\mathcal{C}_{contact}$, a vertex of $\mathcal{S}$, the intersection of two free edges, the intersection of a free edge and $\mathcal{C}_{contact}$, or the intersection of a free edge and $\mathcal{S}$. At each of these events, the algorithm updates the status of the line and the list of future events. During the sweep, the algorithm traces out the backprojection. Sweep stops when the last event is encountered.

The key idea underlying this algorithm is that the backprojection's boundary consists of straight segments and that at every event there exists a local criterion (which does not depend on what will happen later during the sweep) for deciding how to trace the backprojection's boundary. This criterion leads to erect rays (yielding free edges) from both contact and goal vertices, in order to prevent the backprojection from containing configurations from where a motion may reach a vertex or an edge in which it may stick or slide away from the goal with no possibility of returning to it. The algorithm does not require the backprojection to be bounded.

Figures 14 and 15 illustrate and comment on the operations of this sweep-line method on two examples.

Let n be the number of vertices in $\mathcal{C}_{contact}$. Assume the combined complexity of $\mathcal{C}_{contact}$ and $\mathcal{S}$, i.e. the number of vertices in $\mathcal{C}_{contact}$, plus the number of vertices of $\mathcal{S}$, plus the number of intersections between $\mathcal{S}$ and $\mathcal{C}_{contact}$, to be $O(n)$. The sweep-line algorithm erects $O(n)$ rays. The contour of $\mathcal{L}_\theta(\mathcal{S})$ consists of goal edges (edges bounding $\mathcal{S}$), free edges and contact edges. Each erected ray can contribute at most one edge to the backprojection. Each goal edge contributes at most one backprojection edge. Each non goal edge in contact space can contribute one or

TESLA EXHIBIT 1013, Page 482

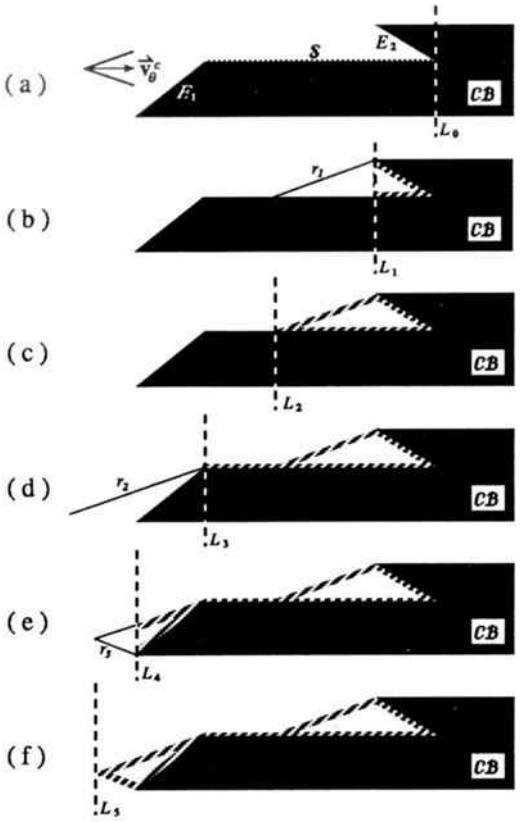


Figure 14. This figure illustrates the computation of a backprojection using the sweep-line algorithm. The goal region is the edge $\mathcal{S}$. Figure a shows the position of the line L when the sweep starts. Figures b through f show its position at the successive critical events. In each figure the striped contour depicts the portion of backprojection constructed so far. The first event (Figure b) occurs when L crosses the non goal vertex bounding the edge E_2. The algorithm erects a ray denoted by r_1 from this vertex. No configurations on the left-hand side of this ray (with the direction of $\vec{\mathbf{v}}_\theta^c$ as the reference) belong to the backprojection of $\mathcal{S}$. The intersection of r_1 with $\mathcal{S}$ produces the next event (Figure c). The following two events are due to the vertices bounding the sliding edge E_1 (Figures d and e). The rays r_2 and r_3 are erected at these vertices. No configurations on the left-hand side of r_2, nor on the right-hand side of r_3, belong to the backprojection. The intersection of the two rays forms the closing event (Figure f).

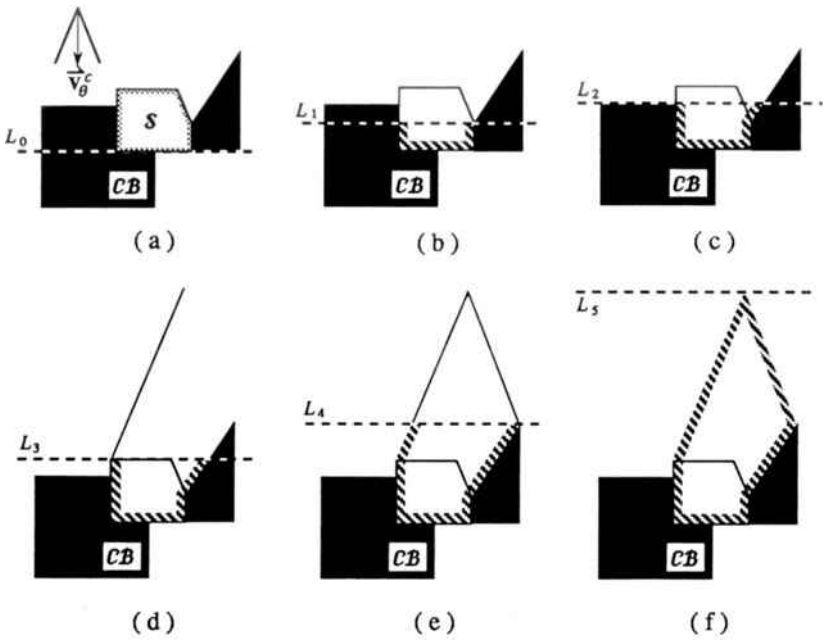


Figure 15. This figure is another illustration of the sweep-line algorithm in the case where $\mathcal{S}$ is a polygonal region in $\mathcal{C}_{valid}$. In Figure d, notice the erection of a ray from a goal vertex located in free space.

several backprojection edges, but whenever it contributes k backprojection edges, it also "consumes" $O(k)$ free edges. Hence, the boundary of the backprojection consists of $O(n)$ edges. Since each ray is interrupted at its first intersection with a goal edge, a contact edge, or another ray, the backprojection's boundary is computed in $O(n \log n)$ time.

4.4 Backprojection from Sticking Edges

Backprojection from sticking edges is a simple method for computing preimages. Given a commanded velocity $\vec{v}_\theta^c$ and a goal $\mathcal{T}$, it consists of:

1. Determining the subset $\mathcal{T}^s$ of $\mathcal{T} \cap \mathcal{C}_{contact}$ in which motions commanded along $\vec{v}_\theta^c$ are guaranteed to stick.

2. Computing the backprojection $\mathcal{L}_\theta(\mathcal{T}^s)$.

TESLA EXHIBIT 1013, Page 484

The computation of $\mathcal{T}^s$ is simply done by comparing the relative orientation of each contact edge in $\mathcal{T}$ with $\vec{\mathbf{v}}_\theta^c$. $\mathcal{T}^s$ is made of every edge E in $\mathcal{T} \cap \mathcal{C}_{contact}$ such that:[12]

$$|angle(-\vec{\mathbf{v}}_\theta^c, \vec{\nu}(E))| \leq \phi - \eta_v.$$

(Notice that guaranteed sticking in an edge is possible only if $\phi > \eta_v$, i.e. if the friction cone is larger than the control uncertainty cone.)

Assume that $\mathcal{T}^s$ is not empty (otherwise, the constructed preimage is the empty set). We still have to construct the termination condition that recognizes sticking. Indeed, while sticking "physically" terminates a motion, the robot must recognize sticking in order to "logically" terminate the motion and execute the next motion command in the motion strategy or return success. One way to construct the termination condition is to project $\mathcal{L}_\theta(\mathcal{T}^s)$ on a line parallel to $\vec{\mathbf{v}}_\theta^c$. Let D be the length of the projection. Since the friction cone is larger than the control uncertainty cone, the actual velocity $\vec{\mathbf{v}}$ of the robot has a positive component along $\vec{\mathbf{v}}_\theta^c$ everywhere in $\mathcal{L}_\theta(\mathcal{T}^s)$, i.e. $\vec{\mathbf{v}} \cdot \vec{\mathbf{v}}_\theta^c > 0$. By examining all the contact edges in the boundary of $\mathcal{L}_\theta(\mathcal{T}^s)$, one can compute the minimal value v_{min} of $\vec{\mathbf{v}} \cdot \vec{\mathbf{v}}_\theta^c$ in $\mathcal{L}_\theta(\mathcal{T}^s)$. A possible termination condition is:

$$\texttt{TC} \equiv t > \frac{D}{v_{min}}.$$

The region $\mathcal{L}_\theta(\mathcal{T}^s)$ is a preimage of $\mathcal{T}$ for this termination condition, which will be referred to in the rest of the chapter as the *sticking termination condition.*

Let $\mathcal{T}$ be input as a list of $O(n)$ contact edges. The computation of $\mathcal{T}^s$ and $\mathcal{L}_\theta(\mathcal{T}^s)$ takes $O(n)$ time and $O(n \log n)$ time, respectively. The construction of the expression TC takes $O(n)$ time.

Figure 16 illustrates a strategy generated by a simple planner which computes preimages using the backprojection from sticking edges method [Latombe, Lazanas and Shekhar, 1989]. The planner constructs and

[12] We do not include isolated sticking vertices in $\mathcal{T}^s$, but we could easily do it. In practice, only the isolated concave sticking vertices are interesting, since the sticking behavior at the other isolated sticking vertices is not stable under small displacements. Sticking at a concave vertex may in fact be easier than sticking along a contact edge; hence, it may yield large preimages.

TESLA EXHIBIT 1013, Page 485

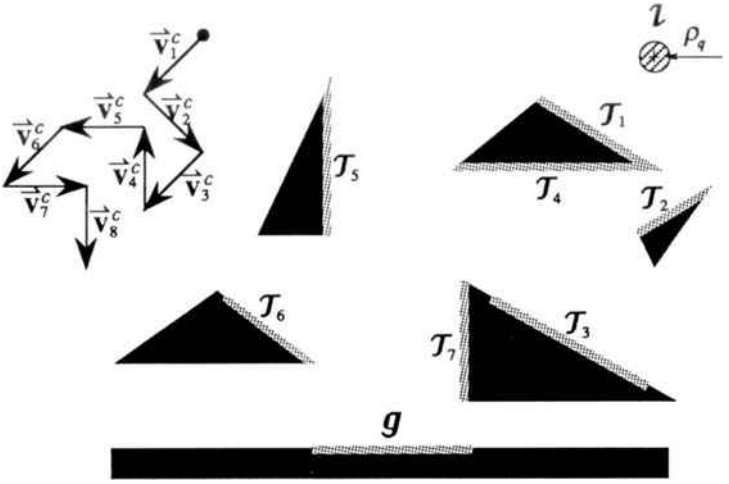


Figure 16. This figure illustrates a strategy produced by a simple planner using the backprojection-from-sticking-edges method. The initial region $\mathcal{I}$ is the disc at the top-right and the goal region is $\mathcal{G}$. The strategy consists of 8 motion commands whose commanded velocities, $\vec{\mathbf{v}}_1^c$ through $\vec{\mathbf{v}}_8^c$, are shown on the left-hand side. These velocities were selected from the set $\{k\pi/4\}_{k=0,\ldots,7}$. Each region shown as a bold grey line segment and denoted by $\mathcal{T}_i$ $(i = 1, \ldots, 7)$ represents the sticking subset (for $\vec{\mathbf{v}}_i^c$) of the preimage of $\mathcal{T}_{i+1}$ $(\mathcal{T}_8 = \mathcal{G})$.

searches a graph of preimages by considering commanded velocities of orientations $\{k\pi/4\}_{k=0,\ldots,7}$. The goal region $\mathcal{G}$ is a subset of a contact edge. The initial region $\mathcal{I}$ is the disc of radius ρ_q shown at the top right of the figure[13]. The generated strategy consists of 8 motion commands. The commanded velocities $\vec{\mathbf{v}}_1^c$ through $\vec{\mathbf{v}}_8^c$ of these commands are depicted on the left-hand side of the figure. $\mathcal{T}_i$ $(i \in [1, 7])$ denotes the sticking subset (for $\vec{\mathbf{v}}_i^c$) of the preimage of $\mathcal{T}_{i+1}$ $(\mathcal{T}_8 = \mathcal{G})$. Hence, a motion along $\vec{\mathbf{v}}_{i+1}^c$ starting from within $\mathcal{T}_i$ terminates by sticking in $\mathcal{T}_{i+1}$. The example of Figure 16 is a typical example where the backprojection-from-sticking-edges method works well. The C-obstacles are such that the robot can "bounce" from one sticking edge to another. In contrast, the method works poorly or fails when the C-obstacles are far apart and when the goal lies mostly (or completely) in free space.

[13]This initial region corresponds to the case where the robot senses its initial configuration before planning.

4.5 Backprojection from Goal Kernel

Backprojection from goal kernel is another method for computing preimages. Given a commanded velocity $\vec{\mathbf{v}}_\theta^c$ and a goal $\mathcal{T}$, it consists of:

1. Identifying a subset $\chi_\theta(\mathcal{T})$ of $\mathcal{T}$ whose achievement by a motion commanded along $\vec{\mathbf{v}}_\theta^c$ is recognizable by a termination condition whose only arguments are $\mathbf{q}^*(t)$ and $\vec{\mathbf{f}}^*(t)$, independent of the starting region.

2. Computing the backprojection $\mathcal{L}_\theta(\chi_\theta(\mathcal{T}))$.

The subset $\chi_\theta(\mathcal{T})$ is called the *θ-kernel* of $\mathcal{T}$.

We say that a configuration $\mathbf{q} \in \mathcal{C}_{valid}$ is **θ-consistent** with $\mathbf{q}^*$ and $\vec{\mathbf{f}}^*$ if and only if, when the robot commanded along $\vec{\mathbf{v}}_\theta^c$ is at configuration $\mathbf{q}$, $\mathbf{q}^*$ and $\vec{\mathbf{f}}^*$ are possible sensed configuration and force given the geometry of $\mathcal{C}_{valid}$, the friction angle ϕ and the uncertainty in sensing and control. The direction of the commanded velocity plays an important role in θ-consistency because the force that is sensed in contact space depends on it. (The robot may be in contact space without sensing any reaction force, if it moves tangentially to a C-obstacle.) We define $\mathcal{K}_\theta(\mathbf{q}^*, \vec{\mathbf{f}}^*)$ as the set of configurations $\mathbf{q}$ that are θ-consistent with $\mathbf{q}^*$ and $\vec{\mathbf{f}}^*$.

Assume for simplification that the uncertainty in the moduli of the commanded velocity and the sensed force is zero[14], i.e. $\Delta_v(\vec{\mathbf{v}}_\theta^c) = \{1\}$ and $\Delta_f(\vec{\mathbf{f}}^*) = \{\|\vec{\mathbf{f}}^*\|\}$. Then:

- If $\|\vec{\mathbf{f}}^*\| = 0$, $\mathcal{K}_\theta(\mathbf{q}^*, \vec{\mathbf{f}}^*)$ consists of all the valid configurations $\mathbf{q}$ which are distant from $\mathbf{q}^*$ by less than ρ_q and lie either in free space, or in contact space at a point where the actual reaction force may be zero:

$$\mathcal{K}_\theta(\mathbf{q}^*, \vec{0}) = \{\mathbf{q} \in \mathcal{C}_{valid} \,/\, \|\mathbf{q} - \mathbf{q}^*\| < \rho_q \,,\, \vec{0} \in F_{reac}(\mathbf{q}, \theta)\}.$$

  ($F_{reac}(\mathbf{q}, \theta)$ is the set of possible reaction forces at $\mathbf{q}$ when the commanded velocity is $\vec{\mathbf{v}}_\theta^c$. See Subsection 1.3.3.)

[14] Removing these assumptions is straightforward, but complicates slightly the construction of $\mathcal{K}_\theta(\mathbf{q}^*, \vec{\mathbf{f}}^*)$.

TESLA EXHIBIT 1013, Page 487

- If $\|\vec{\mathbf{f}}^*\| \neq 0$, $\mathcal{K}_\theta(\mathbf{q}^*, \vec{\mathbf{f}}^*)$ consists of all the contact configurations $\mathbf{q}$ which are distant from $\mathbf{q}^*$ by less than ρ_q and which may generate a non-zero reaction force $\vec{\mathbf{f}}$ whose angle with $\vec{\mathbf{f}}^*$ is less than ε_f:

$$\mathcal{K}_\theta(\mathbf{q}^*, \vec{\mathbf{f}}^*) = \{\mathbf{q} \in \mathcal{C}_{contact} \,/\, \|\mathbf{q} - \mathbf{q}^*\| < \rho_q \;,\; \exists \vec{\mathbf{f}} \neq \vec{0} \in F_{reac}(\mathbf{q}, \theta) : \; |angle(\vec{\mathbf{f}}, \vec{\mathbf{f}}^*)| < \varepsilon_f\}.$$

The termination condition defined as:

$$\mathcal{K}_\theta(\mathbf{q}^*(t), \vec{\mathbf{f}}^*(t)) \subseteq \mathcal{T}$$

is such that when it evaluates to `true` it is guaranteed that the robot is in the goal. Its arguments are $\mathbf{q}^*(t)$ and $\vec{\mathbf{f}}^*(t)$, i.e. the sensed configuration and force at the time t when it is evaluated. Its predicate's knowledge is the geometry of $\mathcal{C}_{valid}$, the commanded velocity, the friction angle ϕ, and the uncertainty in control and in sensing (i.e. $\mathcal{U}_v$, $\mathcal{U}_q$ and $\mathcal{U}_f$). In the rest of the chapter, this condition will be referred to as the *instantaneous sensing termination condition.*

It remains now to define a subset of $\mathcal{T}$ such that if it is attained by a motion, it is guaranteed that the above termination condition evaluates to **true**. This subset is precisely the kernel of the goal.

Let $\mathbf{q}_1$ and $\mathbf{q}_2$ be two configurations in $\mathcal{C}_{valid}$. $\mathbf{q}_1$ and $\mathbf{q}_2$ are said to be θ-**distinguishable** if and only if:

$$\{(\mathbf{q}^*, \vec{\mathbf{f}}^*) \,/\, \mathbf{q}_1, \mathbf{q}_2 \in \mathcal{K}_\theta(\mathbf{q}^*, \vec{\mathbf{f}}^*)\} = \emptyset.$$

In other words, if two configurations $\mathbf{q}_1$ and $\mathbf{q}_2$ are θ-distinguishable, it is guaranteed that during a motion commanded along $\vec{\mathbf{v}}_\theta^c$ there is no instant when the sensed configuration and force are θ-consistent with both $\mathbf{q}_1$ and $\mathbf{q}_2$. Two configurations $\mathbf{q}_1$ and $\mathbf{q}_2$ which are not θ-distinguishable are θ-**confusable.**

The θ-**kernel** of $\mathcal{T}$ is:

$$\chi_\theta(\mathcal{T}) = \{\mathbf{q} \in \mathcal{T} \,/\, \forall \mathbf{q}' \in \mathcal{C}_{valid} \backslash \mathcal{T} : \mathbf{q} \text{ and } \mathbf{q}' \text{ are } \theta\text{-distinguishable}\}.$$

Again notice the importance of θ. Without taking the commanded velocity into account, we would be forced to distinguish between configurations using position sensing only, since no configuration in contact space

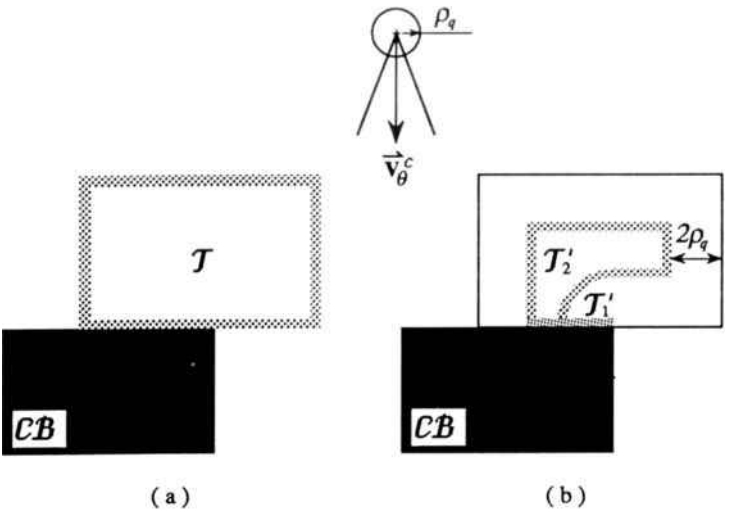


Figure 17. The goal $\mathcal{T}$ is a rectangle in $\mathcal{C}_{valid}$. The commanded velocity $\vec{v}_\theta^c$ points downward. The θ-kernel of $\mathcal{T}$ is $\mathcal{T}_1' \cup \mathcal{T}_2'$. $\mathcal{T}_1'$ is a subset of a contact edge. $\mathcal{T}_2'$ is a generalized polygon.

is guaranteed to generate a non-zero reaction force under all commanded velocities. This would simply lead to isotropically shrinking the goal $\mathcal{T}$ in order to get its kernel.

The computation of $\chi_\theta(\mathcal{T})$ is not difficult (see [Latombe, Lazanas and Shekhar, 1989]). In general, the θ-kernel of a goal consisting of edges in $\mathcal{C}_{contact}$ and polygons in $\mathcal{C}_{valid}$ is a region made of both edges in $\mathcal{C}_{contact}$ and generalized polygons in $\mathcal{C}_{valid}$. Such a kernel is illustrated in Figure 17. In this example, the workspace contains a single rectangular C-obstacle. Figure 17.a displays the goal region $\mathcal{T}$, a rectangle in $\mathcal{C}_{valid}$. The θ-kernel, with $\vec{v}_\theta^c$ pointing downward, is the region $\mathcal{T}_1' \cup \mathcal{T}_2'$ (Figure 17.b). $\mathcal{T}_1'$ is a subset of a contact edge. $\mathcal{T}_2'$ is a generalized polygon. Indeed, a configuration in $\mathcal{T}_1'$ is θ-confusable only with contact configurations contained in the goal. A configuration in $\mathcal{T}_2' \backslash \mathcal{T}_1'$ is θ-confusable only with configurations inside the goal or inside the C-obstacle (but the latter are not achievable). Any configuration outside $\mathcal{T}_1' \cup \mathcal{T}_2'$ is θ-confusable with a configuration in $\mathcal{C}_{valid}$ not located in the goal.

We can compute the backprojection of a region $\chi_\theta(\mathcal{T})$ containing generalized polygons using the Canny-Donald's sweep-line algorithm sketched

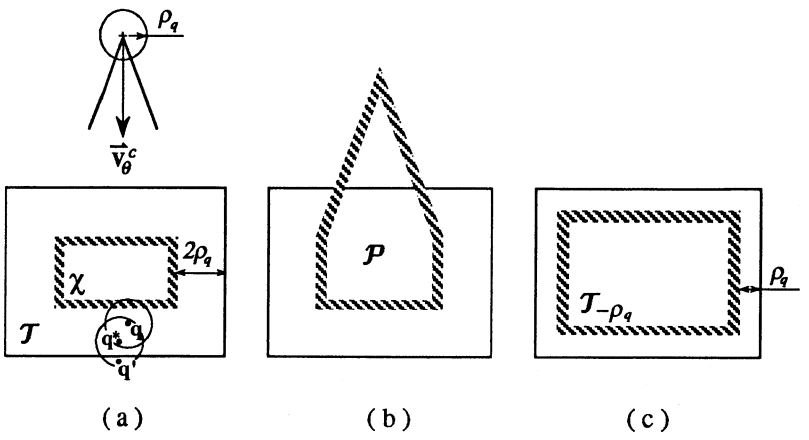


Figure 18. The goal $\mathcal{T}$ is a rectangle in a configuration space empty of C-obstacles. The commanded velocity $\vec{v}_\theta^c$ points along the ingoing normal of the top edge of the rectangle. Figure a shows the kernel χ of $\mathcal{T}$ (striped contour). It is independent of the direction θ since force sensing is useless in the absence of C-obstacles. Figure b shows the preimage $\mathcal{P}$ computed by the backprojection from kernel method. Figure c illustrates the termination condition. The rectangle shown with striped contour depicts the set of sensed configurations at which the termination condition evaluates to `true`.

in Subsection 4.3. The positions of the sweep-line L where it is tangent to the circular edges of $\mathcal{T}$ must be included as additional events. The boundary of the constructed backprojection consists of straight and circular edges. Preimage backchaining may hence lead to computing the kernel of a goal consisting of contact edges and generalized polygons in $\mathcal{C}_{valid}$. One can verify that this kernel also consists of contact edges and generalized polygons in $\mathcal{C}_{valid}$.

The region $\mathcal{L}_\theta(\chi_\theta(\mathcal{T}))$ is the computed preimage of $\mathcal{T}$. Indeed, a motion commanded along $\vec{v}_\theta^c$ from this region is guaranteed to reach $\chi_\theta(\mathcal{T})$ (if it is not terminated before). By definition of $\chi_\theta(\mathcal{T})$, it is then guaranteed that the termination condition $\mathcal{K}_\theta(\mathbf{q}^*(t), \vec{\mathbf{f}}^*(t)) \subseteq \mathcal{T}$ evaluates to `true` at some instant during the motion. The condition certainly evaluates to `true` when $\chi_\theta(\mathcal{T})$ is attained, but it may become `true` before. When the condition becomes `true`, even if the kernel has not been attained yet, the definition of $\mathcal{K}_\theta$ guarantees that the robot is in the goal.

TESLA EXHIBIT 1013, Page 490

Example: Figure 18 shows the preimage computed by this method in the simple case where the goal $\mathcal{T}$ is a rectangle and $\mathcal{C}_{valid} = \mathcal{C}$ (i.e. there is no C-obstacle). The commanded velocity points along the ingoing normal of an edge of the goal. The kernel, which is independent of the commanded velocity (since force sensing is useless), is the rectangle denoted by χ and obtained by "shrinking" $\mathcal{T}$ by $2\rho_q$ as shown in Figure 18.a. (We assume that each of $\mathcal{T}$'s edges is longer than $4\rho_q$.) In order to understand this transformation, consider a configuration $\mathbf{q}$ (see Figure 18.a). This configuration may be sensed as any configuration in the disc of radius ρ_q centered at $\mathbf{q}$. Any sensed configuration $\mathbf{q}^*$ in this disc may be interpreted as a configuration $\mathbf{q}'$ in the disc of radius ρ_q centered at $\mathbf{q}^*$. In order to be sure that this second disc is completely contained in $\mathcal{T}$, the goal has to be shrunk by $2\rho_q$. Figure 18.b shows the computed preimage $\mathcal{P}$ of $\mathcal{T}$. Figure 18.c depicts the set of sensed configurations (the rectangle denoted by $\mathcal{T}_{-\rho_q}$ and obtained by shrinking $\mathcal{T}$ by ρ_q) at which the termination condition $\mathcal{K}(\mathbf{q}^*(t)) \subseteq \mathcal{T}$ evaluates to `true`. (We write $\mathcal{K}_\theta(\mathbf{q}^*(t), \vec{\mathbf{f}}^*(t)) = \mathcal{K}(\mathbf{q}^*(t))$ since, in this particular example, neither the commanded velocity nor force sensing plays any role in the notion of consistency.) ∎

Backprojecting from the goal's θ-kernel usually generates preimages which are significantly larger than those produced by backprojecting from the goal's sticking edges. In particular, it can produce preimages of goals made of regions lying in the free space and/or non-sticking contact edges. There are situations, however, where backprojecting from sticking edges produces larger preimages. Figure 19 shows one of them. The goal $\mathcal{T}$ is a sticking edge for the commanded velocity $\vec{\mathbf{v}}_\theta^c$ and none of the two edges adjacent to $\mathcal{T}$ are distinguishable from $\mathcal{T}$ using force sensing. Hence, the θ-kernel is obtained by shrinking $\mathcal{T}$ by $2\rho_q$ at both its endpoints. The backprojection of the shrunk edge (the θ-kernel) is smaller than the backprojection of the full edge (the sticking edge).

One can combine the two methods by: (1) computing the union S of the sticking edges and the θ-kernel for the selected $\vec{\mathbf{v}}_\theta^c$; and (2) computing the backprojection of S for $\vec{\mathbf{v}}_\theta^c$. This backprojection is the computed preimage $\mathcal{P}$. The termination condition is the disjunction of the two termination conditions given above, i.e. :

$$\left(t > \frac{D}{v_{min}}\right) \bigvee \left(\mathcal{K}_\theta(\mathbf{q}^*(t), \vec{\mathbf{f}}^*(t)) \subseteq \mathcal{T}\right).$$

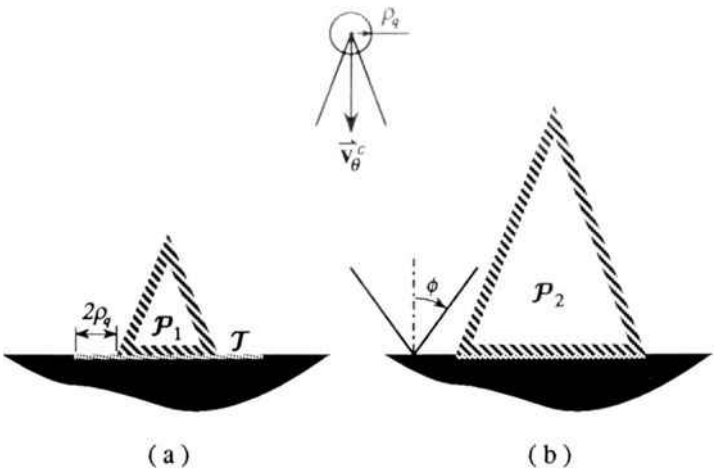


Figure 19. Backprojecting from the goal's θ-kernel often generates preimages which are significantly larger than those produced by backprojecting from the goal's sticking edges. The above figure shows that this is not always true. Figure a displays the preimage $\mathcal{P}_1$ of $\mathcal{T}$ computed by the backprojection-from-kernel method for the commanded velocity $\vec{v}_\theta^c$. Figure b displays the preimage $\mathcal{P}_2$ computed by the backprojection-from-sticking-edges method for the same commanded velocity.

Note that the preimage computed by combining the two methods may be larger than the union of the preimages computed by each of the two methods.

5 Selection of the Commanded Velocity

At this point, we can compute preimages for rather simple termination conditions. In order to plan a motion strategy, we still have to select commanded velocities. As suggested in Section 2, we can construct a tree of preimages by selecting the commanded velocities' orientations from a finite set of values in S^1. Each arc in the tree connects a region to its preimage for a motion commanded along one of these velocities. Figure 16 illustrates a strategy generated by searching such a tree. This naive approach is not very satisfactory, however; indeed, if a sequential motion strategy consisting of motion commands using the sticking or the

TESLA EXHIBIT 1013, Page 492

instantaneous sensing termination condition exists, it may not always be found by the approach.

In this section we present another approach to the selection of the commanded velocities. The approach rests on the observation that when the orientation of the commanded velocity varies over S^1, the topology of a preimage computed as described in the previous section changes at a finite number of critical orientations. We first show how this idea can be used to construct a one-step strategy. Then, we consider the problem of generating a multi-step strategy. We only treat the case of the sticking termination condition. The results can be extended to the instantaneous sensing termination condition by noticing that when θ varies over S^1, the θ-kernel of a goal $\mathcal{T}$ may change only at a finite number of orientations.

5.1 One-Step Strategy

Consider a problem with $\mathcal{I}$ for initial region and $\mathcal{G}$ for goal region. Let $\mathcal{I}$ be a polygon in $\mathcal{C}_{valid}$ and $\mathcal{G}$ a collection of contact edges. Throughout the section, we assume that both $\mathcal{I}$ and $\mathcal{G}$ have constant complexity. We wish to construct a single motion command guaranteed to achieve $\mathcal{G}$ (by sticking in it) from $\mathcal{I}$, if one such command exists. In principle, this requires us to construct the backprojection of the sticking edges in $\mathcal{G}$ for every commanded velocity and to test whether it fully contains $\mathcal{I}$. We show below that this construction is only required at a finite number of critical orientations, where the backprojection topology changes.

5.1.1 Nondirectional Backprojection

DEFINITION 8: *The* **nondirectional backprojection** *of a region $\mathcal{S} \subset \mathcal{C}_{valid}$ is the set:*

$$\mathcal{L}(\mathcal{S}) = \bigcup_{\theta \in S^1} (\mathcal{L}_\theta(\mathcal{S}) \times \{\theta\}) \subset \mathcal{C}_{valid} \times S^1.$$

Let $\mathcal{S}$ consist of a finite collection of contact edges. Consider a backprojection $\mathcal{L}_\theta = \mathcal{L}_\theta(\mathcal{S})$ at some noncritical value of θ. (The notion of critical value of θ will be defined below.) The boundary of $\mathcal{L}_\theta$ consists of contact edges and free edges (see Subsection 4.3). Each free edge E is anchored at a contact vertex and is parallel to one of the two sides of the negated control uncertainty cone. If the commanded velocity's orientation varies

slightly to $\theta + \delta\theta$, so that the interval $[\theta, \theta + \delta\theta]$ contains no critical orientation, the contact edges remain fixed, while the free edges rotate by $\delta\theta$, each around its anchor vertex. Hence, between θ and $\theta + \delta\theta$, the nondirectional backprojection is bounded by algebraic surfaces of two types, *contact surfaces* and *free surfaces*. The contact surfaces are liftings of the corresponding contact edges in $\mathcal{C}_{valid} \times S^1$. Each free surface is a ruled surface generated by a free edge rotating about its anchor vertex.

As we let θ vary over a larger interval, the topology of the backprojection may change at a finite number of orientations θ_i, where one of the following events occurs [Donald, 1988a]:

- A free edge of $\mathcal{L}_{\theta_i}$ ends at a contact vertex. When this event happens, a segment connecting two contact vertices (a link of the visibility graph of $\mathcal{C}_{valid}$) is parallel to a side of the control uncertainty cone.
- The sliding/sticking behavior on a contact edge of $\mathcal{L}_{\theta_i}$ or a contact edge abutting a vertex of $\mathcal{L}_{\theta_i}$ changes at θ_i.
- A vertex of $\mathcal{L}_{\theta_i}$ where two free edges meet lies in a contact edge.

This leads us to define the following **critical orientations**:

- A **vgraph-critical** orientation is an orientation where one side of the control uncertainty cone is parallel to a link of the visibility graph of $\mathcal{C}_{valid}$.
- A **sliding-critical** orientation is an orientation where the sliding/sticking behavior of a contact edge changes.
- A **vertex-critical** orientation is an orientation where the intersection of two rays erected at two contact vertices, each parallel to one side of the control uncertainty cone, lies in a contact edge.

The topology of $\mathcal{L}_\theta$ may change only at these orientations.

There are $O(n^2)$ edges in the visibility graph of $\mathcal{C}_{valid}$ (see Chapter 4). Hence, there are $O(n^2)$ vgraph-critical orientations. The sliding/sticking behavior on a contact edge changes at most four times. Hence, there are $O(n)$ sliding-critical orientations. At every contact vertex, let us erect two rays, each parallel to one side of the control uncertainty cone. These $O(n)$ rays intersect at $O(n^2)$ points. When θ varies over S^1, the

TESLA EXHIBIT 1013, Page 494

various rays rotate together. Therefore, their intersection points describe $O(n^2)$ circles which intersect C_{valid} at $O(n^3)$ points. Hence, there are $O(n^3)$ vertex-critical orientations. In total, there are thus $O(n^3)$ critical orientations.

Let $\theta_{c_1}, \theta_{c_2}, \ldots$ be the critical orientations and θ_{nc_i} be any noncritical orientation in the interval $(\theta_{c_i}, \theta_{c_{i+1}})$. We can represent a nondirectional backprojection as a finite ordered set $\{\mathcal{L}_{\theta_{c_1}}, \mathcal{L}_{\theta_{nc_1}}, \mathcal{L}_{\theta_{c_2}}, \ldots\}$ of backprojections at alternately critical and noncritical orientations. At each noncritical orientation θ_{nc_i}, every edge bounding $\mathcal{L}_{\theta_{nc_i}}$ is labeled according to its type (contact or free edge). Since each backprojection has $O(n)$ bounding edges, the space complexity of this representation is $O(n^4)$. We can compute all the critical orientations in $O(n^3)$ time. For each critical and each noncritical orientation, it takes $O(n \log n)$ time to compute the backprojection. Thus, the representation can be computed in $O(n^4 \log n)$ time.

The above representation and its computation can be significantly improved. First, as noticed in [Donald, 1988a], only $O(n^2)$ vertex-critical orientations need to be considered, yielding a representation of size $O(n^3)$ for the nondirectional backprojection. In addition, we need not compute and represent a full backprojection at each new critical orientation. This has to be done only for one of them. At every other orientation, it suffices to represent the changes of the backprojection with respect to the backprojection at the previous orientation. This yields a representation of the nondirectional backprojection of size $O(n^2)$ which can be incrementally computed in $O(n^2 \log n)$ time [Briggs, 1989].

5.1.2 Application

Assume for a moment that the set $\mathcal{G}^s$ of sticking edges in the goal region $\mathcal{G}$ is constant, i.e. independent of the commanded velocity. This is equivalent to assuming that there exists a termination condition TC that evaluates to **true** if and only if the robot is in $\mathcal{G}^s$.

Under this assumption the nondirectional backprojection $\mathcal{L}(\mathcal{G}^s)$ can be regarded as a *nondirectional preimage* of $\mathcal{G}$ for the condition TC. This means that any slice $\mathcal{L}_\theta(\mathcal{G}^s)$ is a preimage of $\mathcal{G}$ for the motion command $(\vec{\mathbf{v}}^c_\theta, \mathtt{TC})$. The problem of generating a one-step motion strategy can be solved by checking whether there exists an orientation θ such

that $\mathcal{I} \subseteq \mathcal{L}_\theta(\mathcal{G}^s)$ and if there is, returning this orientation. We say that a commanded velocity orientation θ_{g_i} is **initial-region-critical** when a ray parallel to a side of the negated control uncertainty cone erected at a contact vertex contains a vertex of $\mathcal{I}$. Since there are $O(n)$ such rays and $\mathcal{I}$ has constant complexity, there are $O(n)$ initial-region-critical orientations. For each such orientation θ_{g_i}, we can check $\mathcal{L}_{\theta_{g_i}}$ for containment of $\mathcal{I}$ in $O(n)$ time. If there exists an orientation θ_{g_i} such that $\mathcal{I} \subseteq \mathcal{L}_{\theta_{g_i}}$, then a motion commanded along $\vec{\mathrm{v}}^c_{\theta_{g_i}}$ achieves $\mathcal{G}$ from $\mathcal{I}$; otherwise, there is no guaranteed one-step strategy.

Therefore, planning a one-step motion strategy takes $O(n^4 \log n)$ time. This time can be improved to $O(n^2 \log n)$ using Briggs' algorithm for computing nondirectional backprojections.

In reality, the set $\mathcal{G}^s$ of sticking edges in $\mathcal{G}$ is not constant and depends on the orientation θ. However, each edge in $\mathcal{G}$ is guaranteed to be a sticking edge over a single closed connected interval in S^1. By computing these sticking intervals for all the edges in $\mathcal{G}$ and intersecting them, we partition S^1 into a finite set of disjoint (open/closed) intervals $\{(\theta_{s_i}^{min}, \theta_{s_i}^{max})\}_{i=1,2,...}$, such that in each interval $(\theta_{s_i}^{min}, \theta_{s_i}^{max})$ the set $\mathcal{G}_i^s$ of sticking edges in $\mathcal{G}$ remains constant. The nondirectional preimage of $\mathcal{G}$ for the sticking termination condition is:

$$\bigcup_{i=1,2,...} \left(\bigcup_{\theta \in (\theta_{s_i}^{min}, \theta_{s_i}^{max})} (\mathcal{L}_\theta(\mathcal{G}_i^s) \times \{\theta\}) \right).$$

For a goal $\mathcal{G}$ of constant size, there are $O(1)$ intervals $(\theta_{s_i}^{min}, \theta_{s_i}^{max})$, and a representation of size $O(n^4)$ (resp. $O(n^2)$ with the improved algorithm) of the above nondirectional preimage can be computed in $O(n^4 \log n)$ (resp. $O(n^2 \log n)$) time. From this representation, we can compute the initial-region-critical orientations as above and generate a one-step strategy in $O(n^2)$ time.

A slightly more efficient algorithm based on an analysis of the dependence of the problem on the number of connected components in the C-obstacle region has been proposed in [Friedman, Hershberger and Snoeyink, 1990]. This algorithm assumes that $\mathcal{C}_{valid}$ is externally bounded by a polygon. It generates a one-step strategy, if one exists, in $O(kn \log n)$ total time, where k is the number of connected components of the C-obstacle region.

TESLA EXHIBIT 1013, Page 496

It first constructs a data structure of size $O(kn)$ in $O(kn \log n)$ time. From this structure, which is comparable to a nondirectional preimage, it then computes a one-step strategy in anywhere between $O(k \log n)$ and $O(kn)$ time depending on how precisely the initial region $\mathcal{I}$ is described.

5.2 Multi-Step Strategy

In the previous subsection we computed the nondirectional preimage of a goal $\mathcal{G}$ in $\mathcal{C}_{valid} \times S^1$. One way of generalizing this approach to the multi-step case is by introducing the notion of a "multi-step nondirectional preimage" of $\mathcal{G}$ in $\mathcal{C}_{valid} \times S^1 \times ... \times S^1$. Instead of critical values of θ in S^1, we would get critical surfaces in $S^1 \times ... \times S^1$. Although dealing with such surfaces is not impossible a priori, it is certainly not simple.

Another approach to the generation of multi-step strategies has been proposed in [Donald, 1988a]. It makes use of the concept of forward projection introduced below and reduces the problem of planning an r-step motion strategy to a decision problem in the first-order theory of the reals. The problem is described as a Tarski sentence with r existentially quantified variables $\theta_1, \theta_2, ..., \theta_r$ representing the orientations of the commanded velocities in the strategy to be generated. This sentence expresses the facts that the motion along $\vec{\mathbf{v}}^c_{\theta_i}$ starts from where the motion along $\vec{\mathbf{v}}^c_{\theta_{i-1}}$ terminated, or from $\mathcal{I}$ if $i = 1$, and that it terminates by sticking, in $\mathcal{G}$ if $i = r$. Hence, the sentence is built by forward chaining from the initial region.

The *forward projection* of a subset $\mathcal{R}$ of valid space, for the commanded velocity $\vec{\mathbf{v}}^c_\theta$, is the set of all the configurations that may be reached by a motion commanded along $\vec{\mathbf{v}}^c_\theta$ and starting anywhere in $\mathcal{R}$, assuming no termination condition.

DEFINITION 9: *Let $\mathcal{R}$ be a subset of $\mathcal{C}_{valid}$. The* **forward projection** *of $\mathcal{R}$ for the commanded velocity $\vec{\mathbf{v}}^c_\theta$ is the region:*

$$\mathcal{F}_\theta(\mathcal{R}) = \{\tau(t) \;/\; t \in [0, +\infty)\,,\; \tau \in \mathcal{D}_\theta(\mathcal{R})\}$$

where $\mathcal{D}_\theta(\mathcal{R})$ is the directory of actual trajectories for $\mathcal{R}$ and $\vec{\mathbf{v}}^c_\theta$.

The forward projection of a polygonal region $\mathcal{R}$ in $\mathcal{C}_{valid}$ can be computed using a variant of the Canny-Donald sweep-line algorithm presented in

Subsection 4.3 [Donald, 1988a]. The sweep-line L is moved across the plane perpendicularly to the commanded velocity $\vec{\mathbf{v}}^c_\theta$. The sweep starts at a position of L where it is tangent to $\mathcal{R}$, with $\mathcal{R}$ entirely lying on the side of L pointed to by $\vec{\mathbf{v}}^c_\theta$. The sweep proceeds in the direction of $\vec{\mathbf{v}}^c_\theta$. Since the sweep proceeds monotonically in the θ direction, it requires the actual velocity to never project negatively onto the θ direction. This is achieved by imposing that the friction cone be larger than the control uncertainty cone, i.e. $\phi > \eta_v$. (This condition was previously assumed in order to make guaranteed sticking possible.)

Let $\texttt{Stick}_\theta(\mathbf{q})$ be the condition which evaluates to `true` when sticking is possible at $\mathbf{q}$ under a commanded velocity $\vec{\mathbf{v}}^c_\theta$. The set:

$$\mathcal{S}_\theta(\mathcal{R}) = \{\mathbf{q} \in \mathcal{F}_\theta(\mathcal{R}) \;/\; \texttt{Stick}_\theta(\mathbf{q})\}$$

is the set of all configurations in the forward projection of $\mathcal{R}$ where sticking is possible.

Let `Forward` and `StickForward` be two predicates defined as:

$$\begin{aligned} \forall \mathbf{q}_1, \mathbf{q}_2 \in \mathcal{C}_{valid} : \quad & \texttt{Forward}(\theta, \mathbf{q}_1, \mathbf{q}_2) \equiv [\mathbf{q}_2 \in \mathcal{F}_\theta(\{\mathbf{q}_1\})] \\ & \texttt{StickForward}(\theta, \mathbf{q}_1, \mathbf{q}_2) \equiv [\mathbf{q}_2 \in \mathcal{S}_\theta(\{\mathbf{q}_1\})]. \end{aligned}$$

The set $\{(\theta, \mathbf{q}_1, \mathbf{q}_2) \;/\; \texttt{Forward}(\theta, \mathbf{q}_1, \mathbf{q}_2)\}$ is semi-algebraic and it can be described by a polynomial-sized (in the number n of vertices of $\mathcal{C}_{contact}$) quantifier-free polynomial expression of the variables $\theta, x_1, y_1, x_2, y_2$, with $\mathbf{q}_1 = (x_1, y_1)$ and $\mathbf{q}_2 = (x_2, y_2)$. Moreover, this expression can be computed in time polynomial in n [Donald, 1988a]. This result extends to the set $\{(\theta, \mathbf{q}_1, \mathbf{q}_2) \;/\; \texttt{StickForward}(\theta, \mathbf{q}_1, \mathbf{q}_2)\}$, since:

$$\texttt{StickForward}(\theta, \mathbf{q}_1, \mathbf{q}_2) \equiv \texttt{Forward}(\theta, \mathbf{q}_1, \mathbf{q}_2) \wedge \texttt{Stick}_\theta(\mathbf{q}_2).$$

For a given integer r, we define the predicate $r{*}\texttt{StickForward}$ as:

$$\begin{aligned} & r{*}\texttt{StickForward}(\theta_1, ..., \theta_r, \mathbf{q}_0, \mathbf{q}_1, ..., \mathbf{q}_r) \equiv \\ & \qquad \texttt{StickForward}(\theta_1, \mathbf{q}_0, \mathbf{q}_1) \wedge ... \wedge \texttt{StickForward}(\theta_r, \mathbf{q}_{r-1}, \mathbf{q}_r). \end{aligned}$$

The set:

$$\{(\theta_1, ..., \theta_r, \mathbf{q}_0, ..., \mathbf{q}_r) \;/\; r{*}\texttt{StickForward}(\theta_1, ..., \theta_r, \mathbf{q}_0, \mathbf{q}_1, ..., \mathbf{q}_r)\}$$

is a semi-algebraic set that can be described by a polynomial-sized (quantifier-free) polynomial expression.

TESLA EXHIBIT 1013, Page 498

We can now formulate the problem of planning an r-step motion strategy as the problem of deciding the satisfiability of the following Tarski sentence:

$$(\exists \theta_1, ..., \theta_r \in S^1)(\forall \mathbf{q}_0, ..., \mathbf{q}_r \in \mathcal{C}_{valid})$$
$$((\mathbf{q}_0 \in \mathcal{I}) \wedge r*\texttt{StickForward}(\theta_1, ..., \theta_r, \mathbf{q}_0, \mathbf{q}_1, ..., \mathbf{q}_r) \Rightarrow \mathbf{q}_r \in \mathcal{G})$$

and constructing a set of values for θ_1 through θ_r that satisfies this sentence. We could solve this problem using the Collins decomposition, yielding a planning procedure requiring time double-exponential in r. However, another decision procedure given by Grigoryev takes double-exponential time only in the number of quantifier alternations (2 in the above sentence) [Grigoryev, 1988]. Hence, planning an r-step motion strategy can be done in $n^{r^{O(1)}}$ time [Donald, 1988a].

Using a special sort of termination condition, Friedman, Hershberger and Snoeyink proposed a polynomial algorithm for multi-step strategy planning [Friedman, Hershberger and Snoeyink, 1990].

6 Conditional Strategies

So far, we have merely considered the generation of sequential strategies. However, some planning problems only admit conditional strategies for solutions. In this section we study how the search space of the backchaining algorithm should be modified in order to generate such strategies.

Consider the "point-on-hill" example shown in Figure 20.a [Mason, 1984]. The configuration space contains a single, non-compact C-obstacle bounded by three edges, the top edge $\mathcal{G}$, the left edge E_1 and the right edge E_2 (both E_1 and E_2 are semi-infinite lines). The goal region is the top edge $\mathcal{G}$. The initial region $\mathcal{I}$ is all $\mathcal{C}_{valid}$. We assume perfect control, no position sensing, perfect force sensing, and frictionless edges. The problem admits no sequential strategy for solution. However, a motion with a vertical commanded velocity pointing downward and $\|\vec{\mathbf{f}}^*(t)\| > 0$ as the termination condition, is guaranteed to terminate in one of the three edges $\mathcal{G}$, E_1 or E_2. Although it is not possible to know in advance which one of these edges will be achieved, it is known at planning time that this will be decidable at execution time (thanks to force sensing). In addition, it is easy to plan a motion that starts in E_1 (resp. E_2) and terminates in $\mathcal{G}$. Hence, the point-on-hill problem admits a conditional

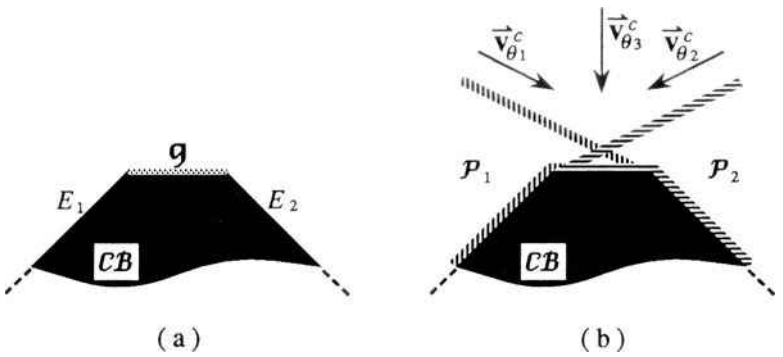


Figure 20. This figure illustrates the "point-on-hill" example. There is a single non-compact C-obstacle bounded by three edges, $\mathcal{G}$, E_1 and E_2 (Figure a). The initial region is $\mathcal{I} = \mathcal{C}_{valid}$ and the goal region is $\mathcal{G}$. We assume perfect control, no position sensing, perfect force sensing, and frictionless edges. The problem admits the following conditional strategy for solution (Figure b). First, move downward along $\vec{v}^c_{\theta_3}$ until a non-zero reaction force is sensed. Then, based on the orientation of this force, decide which one of the edges $\mathcal{G}$, E_1 or E_2 has been achieved. If E_1 (resp. E_2) has been achieved, move along $\vec{v}^c_{\theta_1}$ (resp. $\vec{v}^c_{\theta_2}$) until the orientation of the sensed force is that of the outgoing normal of $\mathcal{G}$.

strategy for solution.

Planning a conditional strategy requires the backchaining process to compute preimages of goal sets of the form $\mathcal{ST} = \{\mathcal{T}_1, ..., \mathcal{T}_s\}$, where $\mathcal{T}_i$ ($i \in [1, s]$) is either the original goal or a previously computed preimage. Let $\mathbf{M} = (\vec{v}^c_\theta, \mathtt{TC})$ be a motion command and $\mathtt{RC}_i$, $i = 1, ..., s$, be conditions called **recognition conditions**. A **preimage** of $\mathcal{ST} = \{\mathcal{T}_1, ..., \mathcal{T}_s\}$ for $\mathbf{M}$ and the $\mathtt{RC}_i$'s is any subset $\mathcal{P}$ of $\mathcal{C}_{valid}$ such that executing $\mathbf{M}$ from any configuration $\mathbf{q}_s \in \mathcal{P}$ is guaranteed to reach $\bigcup_{i=1}^{s} \mathcal{T}_i$ and terminate in it in such a way that, when the motion terminates, at least one recognition condition evaluates to `true` and, if the condition $\mathtt{RC}_i$ (for any $i \in [1, s]$) evaluates to `true`, $\mathcal{T}_i$ has been achieved.

In the example of Figure 20, the backchaining process would ideally proceed as follows:

- First, it would successively generate two preimages of $\mathcal{G}$, $\mathcal{P}_1$ and $\mathcal{P}_2$, for two motions commanded along $\vec{v}^c_{\theta_1}$ and $\vec{v}^c_{\theta_2}$ (see Figure 20.b). Both

motions would have $angle(\vec{\mathbf{f}}^*(t), \vec{\nu}(\mathcal{G})) = 0$ as their termination conditions. Notice that $E_1 \subset \mathcal{P}_1$ and $E_2 \subset \mathcal{P}_2$.

- Second, it would generate $\mathcal{P} = \mathcal{C}_{valid}$ as the preimage of $\{\mathcal{G}, \mathcal{P}_1, \mathcal{P}_2\}$ for a motion commanded along $\vec{\mathbf{v}}^c_{\theta_3}$ (see Figure 20.b), with $\|\vec{\mathbf{f}}^*(t)\| > 0$ as the termination condition. The three conditions $angle(\vec{\mathbf{f}}^*(t), \vec{n}) = 0$, with $\vec{n} = \vec{\nu}(\mathcal{G})$, $\vec{\nu}(E_1)$ and $\vec{\nu}(E_2)$, would be the recognition conditions.

Thus, the generated strategy would be:

move along $\vec{\mathbf{v}}^c_{\theta_3}$ until $\|\vec{\mathbf{f}}^*(t)\| > 0$;
if $angle(\vec{\mathbf{f}}^*(t), \vec{\nu}(\mathcal{G})) = 0$
    then return;
    else if $angle(\vec{\mathbf{f}}^*(t), \vec{\nu}(E_1)) = 0$
        then move along $\vec{\mathbf{v}}^c_{\theta_1}$ until $angle(\vec{\mathbf{f}}^*(t), \vec{\nu}(\mathcal{G})) = 0$;
        else if $angle(\vec{\mathbf{f}}^*(t), \vec{\nu}(E_2)) = 0$
            then move along $\vec{\mathbf{v}}^c_{\theta_2}$ until $angle(\vec{\mathbf{f}}^*(t), \vec{\nu}(\mathcal{G})) = 0$;

The recognition conditions are the conditions labeling the arcs of the tree representing the strategy (see Figure 7).

More generally, in order to generate a conditional plan, the backchaining procedure operates as follows:

1. It creates a set of goals $\mathcal{SG}$ and initializes it to $\{\mathcal{G}\}$.

2. It selects a subset $\mathcal{ST}$ of $\mathcal{SG}$ and computes a preimage $\mathcal{P}$ of $\mathcal{ST}$.

3. If $\mathcal{I} \subseteq \mathcal{P}$, it exits with success. Otherwise, it inserts $\mathcal{P}$ in $\mathcal{SG}$ as a new goal and goes back to step 2.

The preimage of a set of goals $\mathcal{ST}$ can be computed by adapting the methods presented in Section 4. For example, the backprojection-from-goal-kernel method applies as follows. Let $\mathcal{ST} = \{\mathcal{T}_1, ..., \mathcal{T}_s\}$, with $s > 1$, and let $\vec{\mathbf{v}}^c_\theta$ be the selected commanded velocity. The θ-kernel $\chi_\theta(\mathcal{T}_i)$ of every goal $\mathcal{T}_i$, $i = 1, ..., s$, is first computed. A preimage of $\mathcal{ST}$ is then constructed as the backprojection of the union $\bigcup_{i=1}^{s} \chi_\theta(\mathcal{T}_i)$. The recognition condition $\mathtt{RC}_i$ is:

$$\mathcal{K}_\theta(\mathbf{q}^*(t), \vec{\mathbf{f}}^*(t)) \subseteq \mathcal{T}_i$$

for $i = 1$ to s. The termination condition is $\mathtt{TC} \equiv \bigvee_{i=1}^{s} \mathtt{RC}_i$.

A planner can be constructed by restricting the possible orientations of the commanded velocity to a finite set of predetermined orientations. However, the number of goal sets $\mathcal{ST}$ to be considered grows exponentially with the number of preimages generated since the beginning of the backchaining process. Additional techniques for selecting velocities and guiding the choice of goal sets would be needed.

Another empirical method for generating conditional strategies has been proposed[15] by Buckley [Buckley, 1987 and 1989b]. The method consists of partitioning $\mathcal{C}_{valid}$ into non-overlapping regions called *atoms*. A *state* of the robot is a set of atoms (possibly a singleton) so that no two atoms in this set are guaranteed to be distinguishable using sensing. A possible choice for the atoms is the free space and all the contact edges. For every pairs of atoms (A_1, A_2), the planner computes the range of commanded velocities that *may* achieve A_2 from A_1 with the sticking or the instantaneous sensing termination condition (to that purpose, one may use an adaptation of the algorithm given in Subsection 5.1.1 to compute the nondirectional forward projection of a region). Next, it intersects these ranges and constructs the range of commanded velocities that are guaranteed to achieve a state S_2 from a state S_1, for every pair of states S_1 and S_2 (see [Buckley, 1987] for details). The result is an AND/OR graph which remains to be searched for a solution AND subgraph describing a guaranteed strategy. The method assumes that the goal is an atom or a collection of atoms.

7 Power of a Termination Condition

So far, we have considered two relatively simple termination conditions — the sticking and instantaneous sensing termination conditions. There exist more sophisticated conditions which are more powerful at recognizing goal achievement. Such conditions may result in larger preimages, and hence simpler strategies. Some planning problems may also admit solutions only if such conditions are used.

The recognition power of a termination condition depends on the knowledge it uses. This knowledge may include both "dynamic" knowledge —

[15] See [Desai, 1988] and [Desai and Volz, 1989] for a related method.

TESLA EXHIBIT 1013, Page 502

i.e. the arguments (elapsed time and sensing) that the termination predicate can access during execution — and "static" knowledge — i.e. the information that is passed by the planner in the termination predicate. In the following subsections, we describe several termination conditions which illustrate these dependencies.

7.1 Termination Conditions with Initial State

The planner may include the knowledge of the preimage in the termination predicate. Then, at execution time, the termination condition can rule out interpretations of the sensory inputs that suggest configurations of $\mathcal{A}$ which are outside the range of configurations reachable from the preimage, i.e. the forward projection of the preimage (see Subsection 5.2). We formalize this idea and then show its application to two examples.

We say that a configuration $\mathbf{q} \in \mathcal{C}_{valid}$ is θ-$\mathcal{P}$-**consistent** with a sensed configuration $\mathbf{q}^*$ and a sensed force $\vec{\mathbf{f}}^*$ if and only if:

$$\mathbf{q} \in \mathcal{K}_\theta(\mathbf{q}^*, \vec{\mathbf{f}}^*) \cap \mathcal{F}_\theta(\mathcal{P})$$

with $\mathcal{K}_\theta(\mathbf{q}^*, \vec{\mathbf{f}}^*)$ and $\mathcal{F}_\theta(\mathcal{P})$ respectively defined as in Subsections 4.5 and 5.2. This definition captures the fact that if a motion with commanded velocity $\vec{\mathbf{v}}^c_\theta$ is known to begin in a region $\mathcal{P}$, then at any time t during the motion, the current configuration $\mathbf{q}$ is in both $\mathcal{K}_\theta(\mathbf{q}^*(t), \vec{\mathbf{f}}^*(t))$ and $\mathcal{F}_\theta(\mathcal{P})$. We define:

$$\mathcal{K}_{\theta,\mathcal{P}}(\mathbf{q}^*, \vec{\mathbf{f}}^*) \stackrel{\text{def}}{=} \mathcal{K}_\theta(\mathbf{q}^*, \vec{\mathbf{f}}^*) \cap \mathcal{F}_\theta(\mathcal{P}).$$

Two configurations $\mathbf{q}_1$ and $\mathbf{q}_2$ in $\mathcal{C}_{valid}$ are said to be θ-$\mathcal{P}$-**distinguishable** if and only if:

$$\{(\mathbf{q}^*, \mathbf{f}^*) \;/\; \mathbf{q}_1, \mathbf{q}_2 \in \mathcal{K}_{\theta,\mathcal{P}}(\mathbf{q}^*, \mathbf{f}^*)\} = \emptyset.$$

The θ-$\mathcal{P}$-**kernel** of $\mathcal{T}$ is defined as:

$$\chi_{\theta,\mathcal{P}}(\mathcal{T}) \stackrel{\text{def}}{=} \{\mathbf{q} \in \mathcal{T} \;/\; \forall \mathbf{q}' \in \mathcal{C}_{valid} \backslash \mathcal{T} : \mathbf{q} \text{ and } \mathbf{q}' \text{ are } \theta\text{-}\mathcal{P}\text{-distinguishable}\}.$$

When a preimage $\mathcal{P}$ of a goal $\mathcal{T}$ is constructed for a motion command M, it is known, by the definition of the preimage backchaining process, that

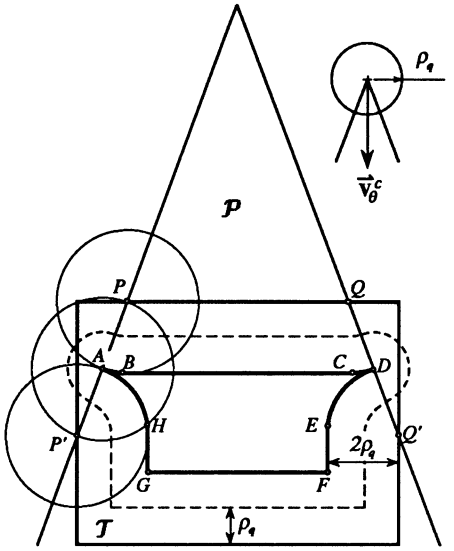


Figure 21. When a termination condition with initial state is used, a preimage of the rectangle $\mathcal{T}$ can be constructed as the backprojection of the region bounded by the generalized polygon whose vertices are A, B, C, D, E, F, G, and H. This preimage is larger than the preimage shown in Figure 18.b.

the execution of M will begin at a starting configuration in $\mathcal{P}$. (Indeed, either $\mathcal{P}$ includes the initial region $\mathcal{I}$, or the backchaining process will recursively find a way to achieve $\mathcal{P}$.) Hence, the region $\mathcal{P}$ defined as the backprojection of the θ-$\mathcal{P}$-kernel of $\mathcal{T}$, i.e. :

$$\mathcal{P} = \mathcal{L}_\theta(\chi_{\theta,\mathcal{P}}(\mathcal{T}))$$

is a preimage of $\mathcal{T}$ for the motion command $(\vec{\mathbf{v}}_\theta^c, \mathtt{TC})$, where:

$$\mathtt{TC} \equiv \mathcal{K}_{\theta,\mathcal{P}}(\mathbf{q}^*(t), \vec{\mathbf{f}}^*(t)) \subseteq \mathcal{T}.$$

This condition is called a termination condition with **initial state.**[16]

[16]This terminology differs from that used in various publications, including [Erdmann, 1984], where a state designates a pair of sensed configuration and force, and the initial state of a motion designates the pair of sensed configuration and force at time 0.

Example 1: Consider the example of Figure 18 again. The goal $\mathcal{T}$ is a rectangle in an empty configuration space. The region whose contour is labeled $ABCDEFGH$ in Figure 21 is a generalized polygon constructed as follows. The straight edge BC is at distance $2\rho_q$ from the top horizontal edge of $\mathcal{T}$. The circular edges AB and CD are circular arcs of radius $2\rho_q$ centered at P and Q, respectively. P (resp. Q) is selected in the top horizontal edge of $\mathcal{T}$ such that the intersection of $\mathcal{T}$ and a line passing through P (resp. Q) and parallel to the left (resp. right) side of the control uncertainty cone is a segment PP' (resp. QQ') of length $4\rho_q$. The circular edges AH and DE are circular arcs of radius $2\rho_q$ with centers at P' and Q', respectively. The straight edge GF is at distance $2\rho_q$ from the bottom edge of $\mathcal{T}$. The straight edges HG and EF are at distance $2\rho_q$ from the left and right edges of $\mathcal{T}$, respectively.

One can verify that the region thus outlined is the kernel $\chi_{\theta,\mathcal{P}}(\mathcal{T})$ with $\mathcal{P} = \mathcal{L}_\theta(\chi_{\theta,\mathcal{P}}(\mathcal{T}))$. In particular, assume that at some instant during the motion the actual configuration is the extreme point marked A in the figure. All the possible sensed configurations at this instant lie in the disc of radius ρ_q centered at A. If the forward projection is not taken into account, the set of all the interpretations of all these possible sensed configurations is the disc of radius $2\rho_q$ centered at A. The intersection of this disc with the forward projection $\mathcal{F}_\theta(\mathcal{P})$ is completely contained in $\mathcal{T}$. Hence, the point A and any configuration outside $\mathcal{T}$ are θ-$\mathcal{P}$-distinguishable, so that A belongs to $\chi_{\theta,\mathcal{P}}(\mathcal{T})$. The same sort of verification can be made with the other vertices B through H, the straight and circular edges connecting these points, and the interior of the outlined area. The resulting preimage is larger that that shown in Figure 18.b. The region outlined in a dotted line depicts the set of sensed configurations $\mathbf{q}^*(t)$ for which the termination condition $\mathcal{K}_{\theta,\mathcal{P}}(\mathbf{q}^*(t)) \subseteq \mathcal{T}$ evaluates to `true` ($\vec{\mathbf{f}}^*(t)$ is useless in this example). ∎

Example 2: Another example is given in Figure 22. The goal region $\mathcal{T}$ is a subset of a contact edge (see Figure 22.a). The preimages $\mathcal{P}_1$ and $\mathcal{P}_2$ which are obtained by backprojecting from the θ-kernel and the θ-$\mathcal{P}_2$-kernel are shown in Figures 22.b and 22.c, respectively. The preimage of Figure 22.c is maximal over all possible termination conditions since it is identical to the maximal backprojection of $\mathcal{T}$. ∎

Notice that there exists one condition of the form:

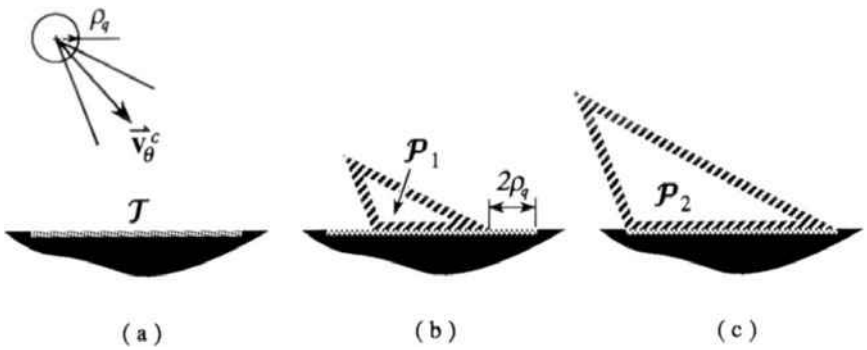


Figure 22. This figure is another illustration of the use of a termination condition with initial state. The goal $\mathcal{T}$ and the commanded velocity are shown in Figure a. The preimage $\mathcal{P}_1$ obtained by backprojecting from the θ-kernel of $\mathcal{T}$ is shown in Figure b. The preimage $\mathcal{P}_2$ obtained by backprojecting from the θ-$\mathcal{P}_2$-kernel of $\mathcal{T}$ is shown in Figure c.

$$\mathcal{K}_{\theta,\mathcal{P}}(\mathbf{q}^*(t), \vec{\mathbf{f}}^*(t)) \subseteq \mathcal{T}$$

for every subset $\mathcal{P}$ of $\mathcal{C}_{valid}$ (in the following we will say that the above expression is a termination condition *schema* parameterized by $\mathcal{P}$). It makes sense to use such a condition as the termination condition of a motion command M only if $\mathcal{P}$ is the preimage in which M's execution starts. The apparent difficulty of using a termination condition with initial state is that the termination condition and therefore the preimage (see Definition 3) are defined in terms of the preimage itself. We will see in Subsection 8.2 that the problem of deciding whether a region $\mathcal{P}$ is a preimage, or not, for a motion command using a termination condition with initial state can be reduced to a decision problem in the first-order theory of the reals. This will be sufficient to define an effective planner capable of using termination conditions with initial state.

7.2 Termination Conditions with Final State

The power of a termination condition may be increased further by embodying the knowledge of itself in the termination predicate. Indeed, the termination condition can then rule out interpretations of the sensory inputs that suggest configurations of $\mathcal{A}$ beyond the limit where the

motion would have been stopped by the termination condition [Latombe, Lazanas and Shekhar, 1989].

Any termination condition TC divides a forward projection $\mathcal{F}_\theta(\mathcal{P})$ into three regions, which we denote by F_1, F_2 and F_3:

- F_1 consists of all the configurations $\mathbf{q} \in \mathcal{F}_\theta(\mathcal{P})$ such that, for every possible sensed configuration $\mathbf{q}^*$ and sensed force $\vec{\mathbf{f}}^*$ θ-consistent with $\mathbf{q}$, TC evaluates to `false`.
- F_2 consists of all the configurations $\mathbf{q} \in \mathcal{F}_\theta(\mathcal{P})$ such that, for every possible sensed configuration $\mathbf{q}^*$ and sensed force $\vec{\mathbf{f}}^*$ θ-consistent with $\mathbf{q}$, TC evaluates to `true`.
- $F_3 = \mathcal{F}_\theta(\mathcal{P}) \backslash (F_1 \cup F_2)$, i.e. any configuration in F_3 may non-deterministically produce sensed configurations and forces for which TC evaluates to either `true` or `false`.

At every instant during a motion starting in $\mathcal{P}$ and commanded along $\vec{\mathbf{v}}_\theta^c$, if the current configuration is in F_1, the motion keeps going; if it is in F_2, the motion stops; if it is in F_3, the motion may either continue or stop. In general, there exist subsets of F_1, F_2 and F_3 that are inaccessible from $\mathcal{P}$ because reaching them would require to previously traverse F_2, where the motion would have been terminated. This is precisely why making the termination condition know itself may increase its recognition power.

Given a termination condition TC, a motion starting in $\mathcal{P}$ and commanded along $\vec{\mathbf{v}}_\theta^c$ can reach a configuration $\mathbf{q}_a$ if and only if there exists a configuration $\mathbf{q}_s \in \mathcal{P}$ and a trajectory $\tau \in \mathcal{D}_\theta(\mathcal{P})$ such that:

- $\tau(0) = \mathbf{q}_s$,
- $\exists t_a \in [0, +\infty) : \tau(t_a) = \mathbf{q}_a$,
- $\forall t \in [0, t_a), \tau(t) \notin F_2$.

Hence, τ connects $\mathbf{q}_s$ to $\mathbf{q}_a$ without traversing F_2, except at $\mathbf{q}_a$ itself (if $\mathbf{q}_a \in F_2$). Let us denote by $\mathcal{F}_{\theta,\mathrm{TC}}(\mathcal{P}) \subseteq \mathcal{F}_\theta(\mathcal{P})$ the set of all the configurations $\mathbf{q}_a$ which can be reached by executing the motion command $\mathbf{M} = (\vec{\mathbf{v}}_\theta^c, \mathrm{TC})$ from within $\mathcal{P}$.

We say that a configuration $\mathbf{q} \in \mathcal{C}_{valid}$ is θ-$\mathcal{P}$-TC-**consistent** with a sensed configuration $\mathbf{q}^*$ and a sensed force $\vec{\mathbf{f}}^*$ if and only if:

$$\mathbf{q} \in \mathcal{K}_\theta(\mathbf{q}^*, \vec{\mathbf{f}}^*) \cap \mathcal{F}_{\theta,\mathrm{TC}}(\mathcal{P}).$$

We define:

$$\mathcal{K}_{\theta,\mathcal{P},\mathrm{TC}}(\mathbf{q}^*, \vec{\mathbf{f}}^*) \stackrel{\text{def}}{=} \mathcal{K}_\theta(\mathbf{q}^*, \vec{\mathbf{f}}^*) \cap \mathcal{F}_{\theta,\mathrm{TC}}(\mathcal{P}).$$

Two configurations $\mathbf{q}_1$ and $\mathbf{q}_2$ in $\mathcal{C}_{valid}$ are said to be θ-$\mathcal{P}$-TC-**distinguishable** if and only if:

$$\{(\mathbf{q}^*, \vec{\mathbf{f}}^*) \,/\, \mathbf{q}_1, \mathbf{q}_2 \in \mathcal{K}_{\theta,\mathcal{P},\mathrm{TC}}(\mathbf{q}^*, \vec{\mathbf{f}}^*)\} = \emptyset.$$

The θ-$\mathcal{P}$-TC-**kernel** of a goal $\mathcal{T}$ is defined as:

$\chi_{\theta,\mathcal{P},\mathrm{TC}}(\mathcal{T}) = \{\mathbf{q} \in \mathcal{T} \,/\, \forall \mathbf{q}' \in \mathcal{C}_{valid} \backslash \mathcal{T} : \mathbf{q}$ and $\mathbf{q}'$ are θ-$\mathcal{P}$-TC-distinguishable$\}$.

A preimage $\mathcal{P}$ of a goal $\mathcal{T}$ is the backprojection of the θ-$\mathcal{P}$-TC-kernel of $\mathcal{T}$, with:

$$\mathcal{K}_{\theta,\mathcal{P},\mathrm{TC}}(\mathbf{q}^*(t), \vec{\mathbf{f}}^*(t)) \subseteq \mathcal{T}$$

as the termination condition of the motion commanded along $\vec{\mathbf{v}}^{\,c}_\theta$. This condition is called a termination condition with initial and **final** states.

Example: We show below that using a termination condition with initial and final states makes it possible to construct a preimage larger than that of Figure 21.

Consider Figure 23. Let R (resp. S) be the point in the top horizontal edge of $\mathcal{T}$ such that the intersection of $\mathcal{T}$ and a line passing through R (resp. S) and parallel to the left (resp. right) side of the control uncertainty cone is a segment RR' (resp. SS') of length $2\rho_q$. The line $R'S'$ that forms the upper portion of the striped contour consists of two circular arcs of radius $2\rho_q$, with respective centers R and S, and a straight segment at distance $2\rho_q$ from the top edge of the goal $\mathcal{T}$. The rest of the striped contour is the lower part of $\mathcal{T}$'s boundary. The region thus outlined is the kernel $\chi_{\theta,\mathcal{P},\mathrm{TC}}(\mathcal{T})$ with:

$$\mathcal{P} = \mathcal{L}_\theta(\chi_{\theta,\mathcal{P},\mathrm{TC}}(\mathcal{T}))$$

and:

TESLA EXHIBIT 1013, Page 508

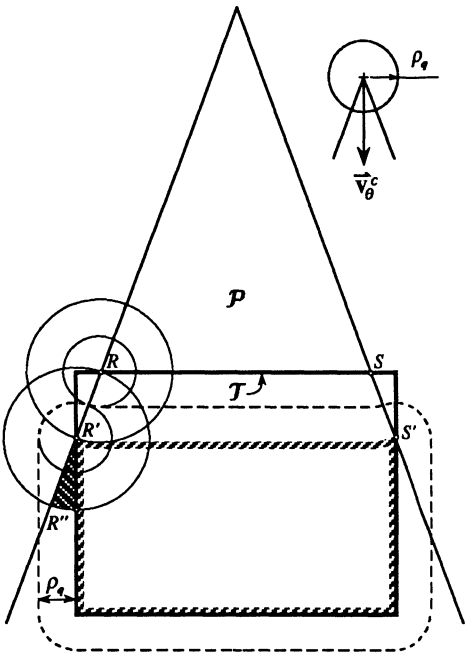


Figure 23. The preimage of the rectangular goal $\mathcal{T}$ for the commanded velocity $\vec{v}_\theta^c$ and the termination condition with initial and final state is the region $\mathcal{P}$ which consists of both the region with a striped outline and the triangle on top of it. The region with a striped outline is the θ-$\mathcal{P}$-TC-kernel of $\mathcal{T}$. The region outlined in a dotted line depicts the set of sensed configurations for which the termination condition $\mathcal{K}_{\theta,\mathcal{P},\mathtt{TC}}(\mathbf{q}^*(t)) \subseteq \mathcal{T}$ evaluates to `true` ($\vec{\mathbf{f}}^*(t)$ is useless).

$$\mathtt{TC} \equiv [\mathcal{K}_{\theta,\mathcal{P},\mathtt{TC}}(\mathbf{q}^*(t)) \subseteq \mathcal{T}]$$

(force sensing is useless). The region outlined in a dotted line depicts the set of sensed configurations $\mathbf{q}^*(t)$ for which the above termination condition evaluates to `true`.

One can verify that the region $\mathcal{P}$ is a preimage of $\mathcal{T}$. Indeed, all the motions starting from $\mathcal{P}$ are guaranteed to reach the goal (if they are not stopped before) and none of them can terminate before it has attained the goal. In addition, none of these motions can leave the goal without terminating.

In order to see how the termination condition uses the knowledge of itself, assume that at some instant during the motion the actual configuration is the point denoted by R'. All the possible sensed configurations at this instant lie in the disc of radius ρ_q centered at R'. If neither the forward projection nor the termination condition are taken into account, the set of all the interpretations of all these sensed configurations is the disc of radius $2\rho_q$ centered at R'. The intersection of this disc with the forward projection $\mathcal{F}_\theta(\mathcal{P})$ contains a sector that is not contained in $\mathcal{T}$. This sector (the striped area in Figure 23) can only be attained from $\mathcal{P}$ by crossing the segment marked $R'R''$. Since for any configuration in this segment, the termination condition evaluates to `true`, the sector cannot be attained. ■

A termination condition with final state is specified as a recursive function of itself. We do not know as yet whether a motion planner using this termination condition can be effectively built.

7.3 Termination Conditions with Time

The modulus of the robot's velocity can only vary in $\Delta_v = (v_1, v_2)$. Hence, at any given instant during the execution of a motion commanded along $\vec{\mathbf{v}}_\theta^c$ and starting from within a region $\mathcal{P}$, the robot is confined to a subset of the forward projection $\mathcal{F}_\theta(\mathcal{P})$. We define the **forward projection of $\mathcal{P}$ at time t** for the commanded velocity $\vec{\mathbf{v}}_\theta^c$ as the region:

$$\mathcal{F}_\theta(\mathcal{P}, t) = \{\tau(t) \;/\; \tau \in \mathcal{D}_\theta(\mathcal{P})\}.$$

We say that two configurations $\mathbf{q}_1$ and $\mathbf{q}_2$ are **θ-$\mathcal{P}$-distinguishable at time t** if and only if:

$$\{(\mathbf{q}^*, \vec{\mathbf{f}}^*) \;/\; \mathbf{q}_1, \mathbf{q}_2 \in \mathcal{K}_\theta(\mathbf{q}^*, \vec{\mathbf{f}}^*) \cap \mathcal{F}_\theta(\mathcal{P}, t)\} = \emptyset.$$

The **θ-$\mathcal{P}$-kernel of $\mathcal{T}$ at time t** is the set of all the configurations in $\mathcal{T}$ which are θ-$\mathcal{P}$-distinguishable at time t from all the configurations in $\mathcal{C}_{valid} \backslash \mathcal{T}$. A preimage can be computed by backprojecting from this kernel. The corresponding termination condition is:

$$\mathcal{K}_\theta(\mathbf{q}^*(t), \vec{\mathbf{f}}^*(t)) \cap \mathcal{F}_\theta(\mathcal{P}, t) \subseteq \mathcal{T}.$$

Example: Consider the point-in-hole example shown in Figure 24. The goal $\mathcal{T}$ is the horizontal contact edge at the bottom of the hole and the

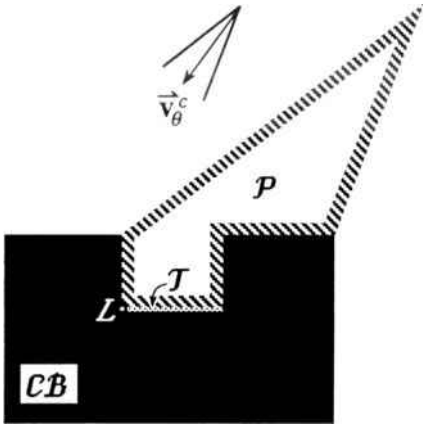


Figure 24. The region $\mathcal{P}$ (striped contour) is the backprojection $\mathcal{L}_\theta(\mathcal{T})$. Any motion starting in $\mathcal{P}$ is guaranteed to reach $\mathcal{T}$ and, after some bounded time T, to stick at the left vertex L of $\mathcal{T}$. Hence, $\mathcal{P}$ is a preimage of $\mathcal{T}$ for the commanded velocity $\vec{\mathbf{v}}_\theta^c$ and the termination condition $t \geq T$.

commanded velocity $\vec{\mathbf{v}}_\theta^c$ points downward toward the left as shown in the figure. Assume no position sensing and no force sensing. (This simply means that $\mathcal{K}_\theta(\mathbf{q}^*, \vec{\mathbf{f}}^*)$ is identical to $\mathcal{C}_{valid}$ in the expressions given above. Hence, it can be removed from them.) Consider the backprojection $\mathcal{P} = \mathcal{L}_\theta(\mathcal{T})$ (shown with striped contour in the figure). Any motion starting in $\mathcal{P}$ is guaranteed to reach $\mathcal{T}$ and, after some bounded time, to stick at the left vertex L of $\mathcal{T}$. Indeed, there exists T such that for all $t \geq T$, $\mathcal{F}_\theta(\mathcal{P}, t) = \{L\}$. Hence, $\mathcal{P}$ is a preimage of $\mathcal{T}$ and a "compiled" form of the termination condition $\mathcal{F}_\theta(\mathcal{P}, t) \subseteq \mathcal{T}$ is $t \geq T$. ∎

The sticking termination condition given in Subsection 4.4 is an instance of a termination condition with time.

7.4 Termination Conditions with Sensing History

A termination condition may have position and force sensing history for argument, rather than just instantaneous sensing (see Subsection 1.3.1). Then it can continuously intersect the set $\mathcal{K}_\theta(\mathbf{q}^*(t), \vec{\mathbf{f}}^*(t))$ and the current forward projection. This intersection is a new region whose forward projection is taken as the current forward projection at the next

instant. At every instant, if the computed intersection is contained in the goal, the termination condition evaluates to **true**.

A procedural (and perhaps more explicit) way of expressing this termination condition consists of continuously pruning the trajectory directory by removing every trajectory τ that is not consistent with current sensing [Mason, 1984]. Then, the termination condition is the following piece of code, which is (idealistically) evaluated at every instant t (D is initialized to $\mathcal{D}_\theta(\mathcal{P})$ at the beginning of the motion, where $\mathcal{P}$ is the preimage from which the motion starts):

for every τ in D do
  if $(\tau(t), \vec{\mathbf{f}}_\tau(t)) \notin \mathcal{U}_q(\mathbf{q}^*(t)) \times \mathcal{U}_f(\vec{\mathbf{f}}^*(t))$ then $D \leftarrow D \backslash \{\tau\}$;
$\mathcal{Q} \leftarrow \{\tau(t) \,/\, \tau \in D\}$;
if $\mathcal{Q} \subseteq \mathcal{T}$ then return **true**; else return **false**;

(This is not an executable algorithm since the set D is usually infinite.)

Example: Consider the motion planning problem depicted in Figure 25. The initial region $\mathcal{I}$ consists of two configurations $\mathbf{q}_{s,1}$ and $\mathbf{q}_{s,2}$. The goal region $\mathcal{G}$ consists of two configurations $\mathbf{q}_{g,1}$ and $\mathbf{q}_{g,2}$. The commanded velocity $\vec{\mathbf{v}}_\theta^c$ is pointing vertically downward. We assume perfect control. Both the distances between $\mathbf{q}_{s,1}$ and $\mathbf{q}_{s,2}$, and between $\mathbf{q}_{g,1}$ and $\mathbf{q}_{g,2}$, are smaller than $2\rho_q$, so that neither the two initial configurations, nor the two goal configurations are guaranteed to be distinguishable at execution time. Since control is perfect, the trajectory directory $\mathcal{D}_\theta(\{\mathbf{q}_{s,1}, \mathbf{q}_{s,2}\})$ consists of two trajectories. The two trajectories are guaranteed to be distinguishable during some interval of time before any of the two goal configurations is attained. Hence, the set $\{\mathbf{q}_{s,1}, \mathbf{q}_{s,2}\}$ is a preimage of $\{\mathbf{q}_{g,1}, \mathbf{q}_{g,2}\}$ for $\mathbf{M} = (\vec{\mathbf{v}}_\theta^c, \mathtt{TC})$, where TC is a termination condition with sensing history. This example does not, however, require continuous sensing history. It could also be solved by generating a conditional strategy in which no motion command uses a termination condition with sensing history. The pertinent "sensing history" is then incorporated into the conditional branching of the strategy. ∎

The termination condition procedurally defined above is quite general — it uses sensing history, initial state, and time — but does not use final state. It is not known as yet whether an effective motion planner using this condition can be constructed.

TESLA EXHIBIT 1013, Page 512

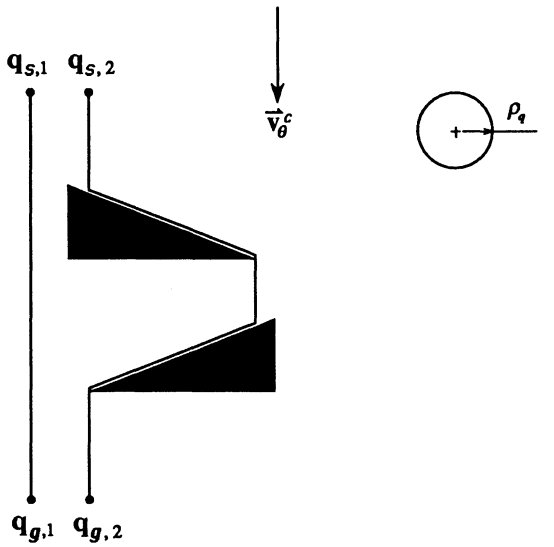


Figure 25. The initial region $\mathcal{I}$ consists of two configurations $q_{s,1}$ and $q_{s,2}$. The goal region $\mathcal{G}$ also consists of two configurations $q_{g,1}$ and $q_{g,2}$. Both the distances between $q_{s,1}$ and $q_{s,2}$, and between $q_{g,1}$ and $q_{g,2}$, are smaller than $2\rho_q$, so that neither the two initial configurations, nor the two goal ones are guaranteed to be distinguishable at execution time. We assume perfect control. The commanded velocity is pointing vertically downward. The trajectory directory $\mathcal{D}_\theta(\{q_{s,1}, q_{s,2}\})$ consists of two trajectories. The two trajectories are guaranteed to be distinguishable during some interval of time. Hence, the set $\{q_{s,1}, q_{s,2}\}$ is a preimage of $\{q_{g,1}, q_{g,2}\}$ for $\mathbf{M} = (\vec{v}_\theta^c, \mathtt{TC})$, with TC being a termination condition with sensing history.

7.5 On Maximal Preimages

The above list of termination conditions illustrates the dependence of the recognition power of a termination condition on the knowledge it uses. Some of the given examples show that in some cases the maximal preimage of a goal over all possible termination conditions may exist. Consider for instance Figure 24. The constructed preimage is identical to the maximal backprojection. Since any preimage is necessarily contained in the maximal backprojection, if a preimage is identical to the maximal backprojection, it is the maximal preimage over all possible termination conditions.

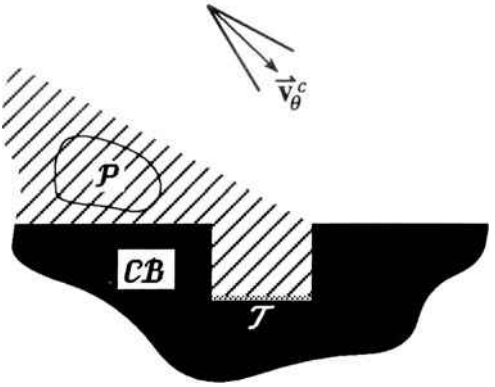


Figure 26. This figure shows a variant of the point-in-hole problem where the C-obstacle is not compact. We assume no sensing. Any bounded subset $\mathcal{P}$ of the striped region is a preimage of $\mathcal{T}$ for a termination condition with time, since any motion starting in $\mathcal{P}$ is guaranteed to stick in the right corner of the hole after some finite amount of time. The striped region, however, is infinite and is not a preimage. Hence, in this example, the goal $\mathcal{T}$ admits no maximal preimage over all possible termination conditions.

Unfortunately, there are also examples showing that the maximal preimage of a goal over all possible termination conditions may not exist, or that, instead, it may exist in quantity. These examples derive from the fact that if $\mathcal{P}_1$ and $\mathcal{P}_2$ are two preimages of the same goal for the same commanded velocity $\vec{v}_\theta^c$, but for two different termination conditions, their union is not a preimage in general. Indeed, the termination condition corresponding to $\mathcal{P}_1$ (resp. $\mathcal{P}_2$) may embody the knowledge of $\mathcal{P}_1$ (resp. $\mathcal{P}_2$), but no termination predicate embodying the knowledge of $\mathcal{P}_1 \cup \mathcal{P}_2$ may be guaranteed to recognize goal achievement by a motion starting from within $\mathcal{P}_1 \cup \mathcal{P}_2$.

Example 1: Consider the point-in-hole problem with a non-compact C-obstacle, as depicted in Figure 26. We assume no sensing. The commanded velocity points downward toward the right as shown in the figure. Every bounded subset $\mathcal{P}$ of the non-compact striped region is a preimage of the edge $\mathcal{T}$, with a termination condition of the form $t > T_\mathcal{P}$. Indeed, by waiting a sufficient amount of time, it is guaranteed that the motion will stick into the bottom-right corner of the hole. However, the

TESLA EXHIBIT 1013, Page 514

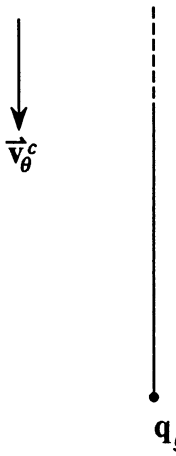


Figure 27. The goal is a single configuration q_g in an empty configuration space. We assume perfect control and no sensing. The commanded velocity is vertical and points downward. Each point in the vertical half-line above q_g is a maximal preimage (for some termination condition with time). Hence, in this example, there exists an infinity of maximal preimages of the goal.

union of all such preimages is the striped region itself, which is not a preimage, since no finite duration T is long enough to guarantee that the goal has been achieved by a motion starting anywhere in the striped region. Hence, there is no maximal preimage. (In this example, the absence of maximal preimage comes from the non-compactness of $C_{contact}$. Another exhibited cause of non-existence of maximal preimage is non-compactness of the goal [Erdmann, 1984].) ∎

Example 2: Consider the point-in-point problem depicted in Figure 27. The goal is a single configuration q_g in an empty configuration space. The commanded velocity is vertical and points downward. Assume perfect control and no position sensing. Each point in the vertical half-line above q_g is a maximal preimage. There is a continuous infinity of them. ∎

If there existed a maximal preimage of a goal over all possible termination conditions, this preimage (assuming it were also computable) could be used as a description of all the other preimages, since any subset of a preimage is also a preimage. Therefore, the possible absence or

multiplicity of maximal preimages (over all termination conditions) is unfortunate because it eliminates one natural representation of all possible preimages. This does not mean, however, that the set of all the preimages of a goal cannot be finitely represented. For instance, in the first example above, the striped region admits a finite description which is also a description of all the preimages of $\mathcal{T}$, since every bounded subset of this region is a preimage. In the second example, the half-line provides a finite description of all the preimages of $\mathbf{q}_g$, since every individual point in this half-line is a preimage.

In Subsection 8.2 we will use a Tarski sentence to finitely describe all the pertinent preimages of a goal and define a very general effective motion planner, despite the possible absence or multiplicity of maximal preimages.

8 Grand Scheme

In Section 2 we gave a simple introduction to preimage backchaining. Then we analyzed important aspects of the approach — preimage computation, choice of commanded velocities, generation of conditional strategies, and power of termination conditions. Now we revisit the approach and we describe the grand preimage backchaining scheme as was originally envisioned in [Lozano-Pérez, Mason and Taylor, 1984]. We first give an abstract planning procedure which formalizes this grand scheme, but ignores computability issues. Next, we present an effective planning method which under some restrictions can generate the same plans as the abstract procedure.

8.1 Abstract Planning Procedure

Let us consider a set of goals $\{\mathcal{T}_\alpha\}$ (goal set) which may possibly be infinite and not even countable. A region $\mathcal{P}$ is a **preimage** of $\{\mathcal{T}_\alpha\}$ for a motion command $\mathbf{M} = (\vec{\mathbf{v}}^{\mathbf{c}}_\theta, \mathtt{TC})$ if and only if any motion starting in $\mathcal{P}$ is guaranteed to terminate inside a member of the goal set $\{\mathcal{T}_\alpha\}$ in such a way that the termination condition TC returns the goal $\mathcal{T}_\alpha$ which has been achieved. (Hence, TC incorporates the recognition conditions introduced in Section 6.)

Let $\mathtt{TC}^{\infty}_{\theta,\mathcal{P},\{\mathcal{T}_\alpha\}}$ be the "most powerful termination condition schema".

This schema, which is parameterized by θ, $\mathcal{P}$ and $\{\mathcal{T}_\alpha\}$, is such that if $\mathcal{P}$ is a preimage of $\{\mathcal{T}_\alpha\}$ for a motion command $(\vec{\mathbf{v}}_\theta^c, \mathtt{TC})$, then it is also a preimage of $\{\mathcal{T}_\alpha\}$ for the motion command $(\vec{\mathbf{v}}_\theta^c, \mathtt{TC}^\infty_{\theta,\mathcal{P},\{\mathcal{T}_\alpha\}})$. Conversely, if $\mathcal{P}$ is not a preimage of $\{\mathcal{T}_\alpha\}$ for the motion command $(\vec{\mathbf{v}}_\theta^c, \mathtt{TC}^\infty_{\theta,\mathcal{P},\{\mathcal{T}_\alpha\}})$, then it is not a preimage of $\{\mathcal{T}_\alpha\}$ for any other motion command $(\vec{\mathbf{v}}_\theta^c, \mathtt{TC})$ with the same value of θ.

Now, let us denote by $\{\mathcal{P}_{\beta_\theta}\}$ the set of all the preimages $\mathcal{P}$ of $\{\mathcal{T}_\alpha\}$ for a motion commanded along $\vec{\mathbf{v}}_\theta^c$ with $\mathtt{TC}^\infty_{\theta,\mathcal{P},\{\mathcal{T}_\alpha\}}$ for termination condition[17]. If $\mathcal{P} \in \{\mathcal{P}_{\beta_\theta}\}$, then every subset of $\mathcal{P}$ is also a preimage, hence is an element of the set $\{\mathcal{P}_{\beta_\theta}\}$.

In its most general form, the preimage backchaining approach is described by the following abstract procedure:

```
procedure PB({T_α});
    begin
        if I ∉ {T_α} then
            begin
                compute {P_βθ} for all θ ∈ S¹;
                PB(⋃_{θ∈S¹} {P_βθ});
            end;
    end;
```

The initial call to this procedure is $\mathrm{PB}(\{\mathcal{G}_\alpha\})$, with $\{\mathcal{G}_\alpha\}$ being the set whose elements are $\mathcal{G}$ and all its subsets. In addition to the above operations, the procedure PB associates the direction θ with every computed preimage $\mathcal{P}_{\beta_\theta}$. (In order to avoid restricting this procedure to planar translational motions of the robot, one can regard θ as an encoding of the commanded velocity in an arbitrary velocity space S^1.)

Assume that the procedure PB terminates after r ($r \geq 0$) recursive calls. Then it has generated a sequence of $r+1$ goal sets, which we denote by $\bigcup_{\theta\in S^1}\{\mathcal{P}^{(k)}_{\beta_\theta}\}$, $k = 0, 1, ..., r$ (k designates the recursion at which the goal set was generated), with $\bigcup_{\theta\in S^1}\{\mathcal{P}^{(0)}_{\beta_\theta}\} = \{\mathcal{G}_\alpha\}$. This sequence determines a conditional strategy to be executed as follows:

1. If $r = 0$, the robot is already in the goal.

[17]This termination condition is different for every preimage $\mathcal{P}$ in the set $\{\mathcal{P}_{\beta_\theta}\}$.

TESLA EXHIBIT 1013, Page 517

2. Otherwise, let $\mathcal{I}_r = \mathcal{I}$; for $k = r, r-1, ..., 1$ do:

   - set θ_k to any of the commanded velocity orientations associated with $\mathcal{I}_k \in \bigcup_{\theta \in S^1} \{\mathcal{P}_{\beta_\theta}^{(k)}\}$;
   - execute the motion command $(\vec{\mathbf{v}}_{\theta_k}^c, \mathrm{TC}^{\infty}_{\theta_k, \mathcal{I}_k, \bigcup_{\theta \in S^1} \{\mathcal{P}_{\beta_\theta}^{(k-1)}\}})$;
   - when the motion terminates, set $\mathcal{I}_{k-1}$ to the goal of $\bigcup_{\theta \in S^1} \{\mathcal{P}_{\beta_\theta}^{(k-1)}\}$ whose achievement has been recognized by the termination condition.

The procedure PB is obviously not an executable algorithm, since the goal sets which are considered and constructed at each recursion are typically noncountable. In addition, the strategies that it generates make use of the termination condition $\mathrm{TC}^{\infty}_{\theta, \mathcal{P}, \{\mathcal{I}_\alpha\}}$, which may not be computable. The fact that the procedure PB described above is not directly implementable does not mean, however, that one cannot define an effective algorithm capable of generating the same plans as PB.

8.2 Planning Method

In this subsection we describe an effective planning method that can generate "almost" the same motion strategies as the abstract procedure PB. This method, which is due to Canny [Canny, 1989], consists of formulating the planning problem as a decision problem in the first-order theory of the reals. It imposes two restrictions: first, it requires the forward projection of any configuration at any time to have an algebraic description; second, it uses a termination predicate that is less powerful than $\mathrm{TC}^{\infty}_{\theta, \mathcal{P}, \{\mathcal{I}_\alpha\}}$. Under these two restrictions, the method is complete and generates an r-step strategy in $O(2^{2^r})$ time whenever one such strategy exists (otherwise it terminates in the same time and returns failure). The double-exponential dependence comes from the Collins decomposition that is applied to eliminate the quantifiers from the Tarski sentence describing the r-step strategy. It is not known as yet whether there exists an effective planning method having the full power of the abstract procedure PB.

8.2.1 Feasible Trajectories

Let us consider a motion commanded along the velocity $\vec{\mathbf{v}}_\theta^c$. We assume that rather than varying continuously, the effective commanded velocity

during the motion takes p successive values $\vec{\mathbf{v}}_\theta^{(1)}, ..., \vec{\mathbf{v}}_\theta^{(p)}$ in $\mathcal{U}_v(\vec{\mathbf{v}}_\theta^c)$, where p is a fixed prespecified integer. Let $t^{(k)}$, $k = 1, ..., p-1$, denote the times at which the effective control velocity switches from $\vec{\mathbf{v}}_\theta^{(k)}$ to $\vec{\mathbf{v}}_\theta^{(k+1)}$. This assumption is aimed at meeting the first restriction mentioned above (algebraicity of the forward projections). It is certainly not the only one possible, but it is a simple one.

Under this assumption, let $\tau : t \in [0, +\infty) \mapsto \tau(t) \in \mathcal{C}_{valid}$ be an actual trajectory that can be generated by a motion commanded along $\vec{\mathbf{v}}_\theta^c$. This trajectory consists of straight segments. The actual velocity is constant along each of these segments. The robot shifts from one segment to the next at each of the instants $t^{(1)}, ..., t^{(p-1)}$, and also when it switches from free to contact space, and conversely. The switch points from free to contact space and from contact to free space are algebraically defined by the parameters $\vec{\mathbf{v}}_\theta^{(1)}, ..., \vec{\mathbf{v}}_\theta^{(p)}, t^{(1)}, ..., t^{(p-1)}$, and by the algebraic description of $\mathcal{C}_{valid}$. There are at most pn such points, where n is the number of contact edges.

Let $\mathbf{s}_\tau(t)$ denote the pair $(\tau(t), \vec{\mathbf{f}}_\tau(t))$, where $\vec{\mathbf{f}}_\tau(t)$ is the reaction force exerted on the robot at time t when it moves along the trajectory τ (see Section 3). This force is algebraically determined by the configuration and the effective commanded velocity at time t (see Subsection 1.3.2).

Let $\overline{\mathcal{D}}_\theta(\{\mathbf{s}_0\})$ be the directory[18] of all feasible actual trajectories τ such that $\mathbf{s}_\tau(0) = \mathbf{s}_0$, which can be generated with the commanded velocity $\vec{\mathbf{v}}_\theta^c$. Each element of $\overline{\mathcal{D}}_\theta(\{\mathbf{s}_0\})$ is a list $(\vec{\mathbf{v}}_\theta^{(1)}, ..., \vec{\mathbf{v}}_\theta^{(p)}, t^{(1)}, ..., t^{(p-1)})$ in:

$$\{(\vec{\mathbf{v}}_\theta^{(1)}, ..., \vec{\mathbf{v}}_\theta^{(p)}, t^{(1)}, ..., t^{(p-1)}) \,/\, \vec{\mathbf{v}}_\theta^{(1)}, ..., \vec{\mathbf{v}}_\theta^{(p)} \in \mathcal{U}_v(\vec{\mathbf{v}}_\theta^c)\ ;\ t^{(1)}, ..., t^{(p-1)} \in [0, +\infty)\ ;\ t^{(1)} \leq t^{(2)} \leq ... \leq t^{(p-1)}\}$$

which uniquely determines an actual trajectory τ such that $(\tau(0), \vec{\mathbf{f}}_\tau(0)) = \mathbf{s}_0$. $\tau(t)$ is defined as an algebraic expression involving both these $2p-1$ parameters and the variable t.

We now define the forward projection of $\mathbf{s}_0$ at time t as:

$$\overline{\mathcal{F}}_\theta(\{\mathbf{s}_0\}, t) = \bigcup_{\tau \in \overline{\mathcal{D}}_\theta(\{\mathbf{s}_0\})} \mathbf{s}_\tau(t).$$

[18] We denote this directory by $\overline{\mathcal{D}}_\theta$ rather than $\mathcal{D}_\theta$ (see Section 3) in order to indicate that the argument of $\overline{\mathcal{D}}_\theta$ is a set of configuration-force pairs, while the argument of $\mathcal{D}_\theta$ is a set of configurations.

TESLA EXHIBIT 1013, Page 519

For any region $\mathcal{X} \subseteq \mathcal{C}_{valid}$, both $\widetilde{\mathcal{X}}_\theta$ and $\widetilde{(\mathcal{X})}_\theta$ denote the set defined as:

$$\widetilde{\mathcal{X}}_\theta = \widetilde{(\mathcal{X})}_\theta = \{(\mathbf{q}, \vec{\mathbf{f}}) \,/\, \mathbf{q} \in \mathcal{X}; \vec{\mathbf{f}} \in F_{reac}(\mathbf{q}, \theta)\}$$

where $F_{reac}(\mathbf{q}, \theta)$ is the set of all the possible reaction forces exerted on the robot at configuration $\mathbf{q}$ when it is commanded along $\vec{\mathbf{v}}^c_\theta$ (see Subsection 1.3.3).

Given a region $\mathcal{R}$ in $\mathcal{C}_{valid}$, we define:

$$\overline{\mathcal{D}}_\theta(\widetilde{\mathcal{R}}_\theta) = \bigcup_{\mathbf{s}_0 \in \widetilde{\mathcal{R}}_\theta} \overline{\mathcal{D}}_\theta(\{\mathbf{s}_0\})$$

and:

$$\overline{\mathcal{F}}_\theta(\widetilde{\mathcal{R}}_\theta, t) = \bigcup_{\mathbf{s}_0 \in \widetilde{\mathcal{R}}_\theta} \overline{\mathcal{F}}_\theta(\{\mathbf{s}_0\}, t) = \bigcup_{\tau \in \overline{\mathcal{D}}_\theta(\widetilde{\mathcal{R}}_\theta)} \mathbf{s}_\tau(t).$$

If $\mathcal{R}$ is a semi-algebraic set, so are $\overline{\mathcal{D}}_\theta(\widetilde{\mathcal{R}}_\theta)$ and $\overline{\mathcal{F}}_\theta(\widetilde{\mathcal{R}}_\theta, t)$.

The only requirements for the method described below are that: (1) every feasible trajectory τ be such that, for every t, $\tau(t)$ is defined by an algebraic expression of t and a finite number of parameters, and (2) $\mathcal{D}_\theta(\{\mathbf{s}_0\})$, for every $\mathbf{s}_0$, be a semi-algebraic subset of this parameter space. Restricting the effective commanded velocity to a finite number of successive values is one specific way to achieve these requirements.

8.2.2 Termination Condition

Given a set $\mathcal{Y}$, each element of which is a pair configuration-force, we denote by $\mathcal{Q}[\mathcal{Y}]$ the set defined as $\mathcal{Q}[\mathcal{Y}] = \{\mathbf{q} \,/\, \exists \vec{\mathbf{f}} : (\mathbf{q}, \vec{\mathbf{f}}) \in \mathcal{Y}\}$. In particular, for any $\mathcal{R} \subseteq \mathcal{C}_{valid}$, we have that: $\mathcal{Q}[\mathcal{R}_\theta] = \mathcal{R}$.

Let $\mathcal{T}$ be a goal described as a semi-algebraic set. Let $\vec{\mathbf{v}}^c_\theta$ be the commanded velocity of a motion intended to achieve this goal from a region $\mathcal{P}$ also described as a semi-algebraic set. This command uses:

$$\mathcal{Q}[\overline{\mathcal{F}}_\theta(\widetilde{\mathcal{P}}_\theta \cap \mathcal{U}_s(\mathbf{s}^*(0)), t) \cap \mathcal{U}_s(\mathbf{s}^*(t))] \subseteq \mathcal{T}$$

as the termination condition, where:

- $\mathbf{s}^*(t) = (\mathbf{q}^*(t), \vec{\mathbf{v}}^c(t))$ represents the pair of sensed configuration and force at time t,

- $\mathcal{U}_s(\mathbf{s}^*) = \mathcal{U}_q(\mathbf{q}^*) \times \mathcal{U}_f(\vec{\mathbf{f}}^*)$, with $\mathbf{s}^* = (\mathbf{q}^*, \vec{\mathbf{f}}^*)$.

This termination condition is less powerful than $\mathrm{TC}^{\infty}_{\theta,\mathcal{P},\{\mathcal{T}_\alpha\}}$ since it uses neither continuous sensing history, nor final state. It nevertheless uses initial state, time, and sensing at times 0 and t, which makes it quite powerful.

From now on, we denote the set $\overline{\mathcal{F}}_\theta(\widetilde{\mathcal{P}}_\theta \cap \mathcal{U}_s(\mathbf{s}^*(0)), t)$ by $\overline{\mathcal{F}}_{\mathcal{P},\mathbf{s}^*(0),\theta}(t)$.

The region $\mathcal{P}$ is a preimage of $\mathcal{T}$ for the motion command specified above if and only if the following *preimage condition* holds:

$$\begin{array}{c} \forall \mathbf{s}_1 \in \widetilde{\mathcal{P}}_\theta \ , \ \forall \mathbf{s}_1^* \in \mathcal{U}_s(\mathbf{s}_1) \ , \ \forall \tau \ , \ \exists t \geq 0 \ , \ \forall \mathbf{s}_1'^* : \\ (\tau \in \overline{\mathcal{D}}_\theta(\widetilde{\mathcal{P}}_\theta \cap \mathcal{U}_s(\mathbf{s}_1^*)) \ \wedge \mathbf{s}_1'^* \in \mathcal{U}_s(\mathbf{s}_\tau(t))) \Rightarrow \mathcal{Q}[\overline{\mathcal{F}}_{\mathcal{P},\mathbf{s}_1^*,\theta}(t) \cap \mathcal{U}_s(\mathbf{s}_1'^*)] \subseteq \mathcal{T}. \end{array} \tag{2}$$

This is a Tarski sentence, and therefore we can decide whether $\mathcal{P}$ is a preimage of $\mathcal{T}$.

This result can be applied to the generation of a one-step motion strategy. Given an initial region $\mathcal{I}$ and a goal region $\mathcal{G}$, both described as semi-algebraic sets, there exists a single motion of commanded velocity $\vec{\mathbf{v}}^c_\theta$ that achieves $\mathcal{G}$ from $\mathcal{I}$ if and only if the sentence:

$$\begin{array}{c} \forall \mathbf{s}_1 \in \widetilde{\mathcal{I}}_\theta \ , \ \forall \mathbf{s}_1^* \in \mathcal{U}_s(\mathbf{s}_1) \ , \ \forall \tau \ , \ \exists t \geq 0 \ , \ \forall \mathbf{s}_1'^* : \\ (\tau \in \overline{\mathcal{D}}_\theta(\widetilde{\mathcal{I}}_\theta \cap \mathcal{U}_s(\mathbf{s}_1^*)) \ \wedge \mathbf{s}_1'^* \in \mathcal{U}_s(\mathbf{s}_\tau(t))) \Rightarrow \mathcal{Q}[\overline{\mathcal{F}}_{\mathcal{I},\mathbf{s}_1^*,\theta}(t) \cap \mathcal{U}_s(\mathbf{s}_1'^*)] \subseteq \mathcal{G} \end{array}$$

is satisfiable. We can eliminate all the quantifiers using the Collins decomposition. It remains a polynomial expression in θ whose solutions are all the guaranteed one-step strategies. The decomposition also gives a sample value of θ.

But, in order to generate multi-step strategies, we still lack a first-order sentence that describes all the possible preimages of $\mathcal{G}$ so that they can be passed recursively as intermediate goals to the planner, as suggested by the procedure PB presented in the previous subsection. A major idea introduced by Canny [Canny, 1989] is that it is not necessary to consider all these preimages. Indeed, only those which can be attained recognizably from the initial region $\mathcal{I}$ by some motion strategy have to be considered. These preimages are called **recognizable preimages**. As shown below, they can be determined by forward chaining from $\mathcal{I}$. It

TESLA EXHIBIT 1013, Page 521

turns out that under the assumptions made above the set of regions that can be recognizably achieved from $\mathcal{I}$ can be algebraically described.

8.2.3 Two-Step Strategy

Consider the problem of planning a two-step strategy to achieve a semi-algebraic goal region $\mathcal{G}$ from a semi-algebraic initial region $\mathcal{I}$. Let $\vec{\mathbf{v}}^c_{\theta_1}$ and $\vec{\mathbf{v}}^c_{\theta_2}$ denote the commanded velocities of the first and the second step, respectively.

Assume for an instant that we have constructed the set $\{\mathcal{P}_{\beta_{\theta_2}}\}$ of all the preimages of $\mathcal{G}$ for all the directions $\theta_2 \in S^1$. We derive from the preimage condition (2) that in order to achieve some preimage $\mathcal{P} \in \{\mathcal{P}_{\beta_{\theta_2}}\}$ from $\mathcal{I}$ with a motion commanded along θ_1, we must have that:

$$\begin{array}{c}\forall \mathbf{s}_1 \in \widetilde{\mathcal{I}}_{\theta_1}\ ,\ \forall \mathbf{s}_1^* \in \mathcal{U}_s(\mathbf{s}_1)\ ,\ \forall \tau_1\ ,\ \exists t_1 \geq 0\ ,\ \forall \mathbf{s}_1'^*\ ,\ \exists \mathcal{P} \in \{\mathcal{P}_{\beta_{\theta_2}}\} : \\ (\tau_1 \in \overline{\mathcal{D}}_{\theta_1}(\widetilde{\mathcal{I}}_{\theta_1} \cap \mathcal{U}_s(\mathbf{s}_1^*)) \ \wedge \mathbf{s}_1'^* \in \mathcal{U}_s(\mathbf{s}_{\tau_1}(t_1))) \Rightarrow \mathcal{Q}[\overline{\mathcal{F}}_{\mathcal{I},\mathbf{s}_1^*,\theta_1}(t_1) \cap \mathcal{U}_s(\mathbf{s}_1'^*)] \subseteq \mathcal{P}.\end{array} \tag{3}$$

This expression is not a first-order sentence since the variable $\mathcal{P}$ does not denote a real, but rather a preimage of $\mathcal{G}$. The satisfiability of such a second-order sentence is known to be undecidable.

Now, given the commanded velocity $\vec{\mathbf{v}}^c_{\theta_1}$, the smallest regions whose achievement from $\mathcal{I}$ can be recognized by the termination condition at time t_1 are regions of the form $\mathcal{Q}[\overline{\mathcal{F}}_{\mathcal{I},\mathbf{s}^*(0),\theta_1}(t_1) \cap \mathcal{U}_s(\mathbf{s}^*(t_1))]$. There is no gain in considering a non-recognizable preimage of $\mathcal{G}$. Hence, we can rewrite the sentence (3) into the following first-order sentence:

$$\begin{array}{c}\forall \mathbf{s}_1 \in \widetilde{\mathcal{I}}_{\theta_1}\ ,\ \forall \mathbf{s}_1^* \in \mathcal{U}_s(\mathbf{s}_1)\ ,\ \forall \tau_1\ ,\ \exists t_1 \geq 0\ ,\ \forall \mathbf{s}_1'^* : \\ (\tau_1 \in \overline{\mathcal{D}}_{\theta_1}(\widetilde{\mathcal{I}}_{\theta_1} \cap \mathcal{U}_s(\mathbf{s}_1^*)) \ \wedge \mathbf{s}_1'^* \in \mathcal{U}_s(\mathbf{s}_{\tau_1}(t_1))) \Rightarrow P_1(\mathbf{s}_1^*, \theta_1, t_1, \mathbf{s}_1'^*)\end{array}$$

where $P_1(\mathbf{s}_1^*, \theta_1, t_1, \mathbf{s}_1'^*)$ is an expression that is satisfied when the region $\mathcal{Q}[\overline{\mathcal{F}}_{\mathcal{I},\mathbf{s}_1^*,\theta_1}(t_1) \cap \mathcal{U}_s(\mathbf{s}_1'^*)]$ is a preimage of $\mathcal{G}$, hence a recognizable preimage. This expression can be constructed from the preimage condition (2):

$$\begin{array}{l}P_1(\mathbf{s}_1^*, \theta_1, t_1, \mathbf{s}_1'^*) \equiv \\ \exists \theta_2 \in S^1\ ,\ \forall \mathbf{s}_2 \in \widetilde{(K_1)}_{\theta_2}\ ,\ \forall \mathbf{s}_2^* \in \mathcal{U}_s(\mathbf{s}_2)\ ,\ \forall \tau_2\ ,\ \exists t_2 \geq 0\ ,\ \forall \mathbf{s}_2'^* : \\ \quad (\tau_2 \in \overline{\mathcal{D}}_{\theta_2}(\widetilde{(K_1)}_{\theta_2} \cap \mathcal{U}_s(\mathbf{s}_2^*)) \ \wedge \ \mathbf{s}_2'^* \in \mathcal{U}_s(\mathbf{s}_{\tau_2}(t_2))) \\ \quad\quad \Rightarrow \mathcal{Q}[\overline{\mathcal{F}}_{K_1,\mathbf{s}_2^*,\theta_2}(t_2) \cap \mathcal{U}_s(\mathbf{s}_2'^*)] \subseteq \mathcal{G}\end{array}$$

TESLA EXHIBIT 1013, Page 522

where $H_1 = \overline{\mathcal{F}}_{\mathcal{I},\mathbf{s}_1^*,\theta_1}(t_1) \cap \mathcal{U}_s(\mathbf{s}_1'^*)$ and $K_1 = \mathcal{Q}[H_1]$.

This leads us to describe the existence of a two-step motion strategy by the following Tarski sentence:

$$\begin{array}{l}\exists\theta_1 \in S^1 \ , \ \forall \mathbf{s}_1 \in \widetilde{\mathcal{I}}_{\theta_1} \ , \ \forall \mathbf{s}_1^* \in \mathcal{U}_s(\mathbf{s}_1) \ , \ \forall \tau_1 \ , \ \exists t_1 \geq 0 \ , \ \forall \mathbf{s}_1'^* : \\ \quad (\tau_1 \in \overline{\mathcal{D}}_{\theta_1}(\widetilde{\mathcal{I}}_{\theta_1} \cap \mathcal{U}_s(\mathbf{s}_1^*)) \ \wedge \ \mathbf{s}_1'^* \in \mathcal{U}_s(\mathbf{s}_{\tau_1}(t_1))) \\ \quad\quad \Rightarrow \exists\theta_2 \in S^1 \ , \ \forall \mathbf{s}_2 \in \widetilde{(K_1)}_{\theta_2} \ , \ \forall \mathbf{s}_2^* \in \mathcal{U}_s(\mathbf{s}_2) \ , \ \forall \tau_2 \ , \ \exists t_2 \geq 0 \ , \ \forall \mathbf{s}_2'^* : \\ \quad\quad\quad (\tau_2 \in \overline{\mathcal{D}}_{\theta_2}(\widetilde{(K_1)}_{\theta_2} \cap \mathcal{U}_s(\mathbf{s}_2^*)) \ \wedge \ \mathbf{s}_2'^* \in \mathcal{U}_s(\mathbf{s}_{\tau_2}(t_2))) \\ \quad\quad\quad\quad \Rightarrow \mathcal{Q}[\overline{\mathcal{F}}_{K_1,\mathbf{s}_2^*,\theta_2}(t_2) \cap \mathcal{U}_s(\mathbf{s}_2'^*)] \subseteq \mathcal{G}.\end{array}$$

We remove the first quantifier $\exists\theta_1 \in S^1$ and perform quantifier elimination on this sentence, yielding all the possible orientations of the commanded velocity for the first step of the strategy. If this set is not empty, i.e. if the above sentence is satisfiable, one value θ_1 is arbitrarily selected and the motion command $(\vec{\mathbf{v}}_{\theta_1}^c, P_1(\mathbf{s}_1^*, \theta_1, t, \mathbf{s}^*(t)))$ is executed, with $\mathbf{s}_1^*$ being the pair of sensed configuration and force at time $t = 0$ (of the first motion). Let t_1 be the instant when the termination condition $P_1(\mathbf{s}_1^*, \theta_1, t, \mathbf{s}^*(t))$ evaluates to `true`. At this instant, the motion has achieved a recognized preimage of $\mathcal{G}$. We remove the first quantifier $\exists\theta_2 \in S^1$ from the expression defining $P_1(\mathbf{s}_1^*, \theta_1, t_1, \mathbf{s}_1'^*)$ and we replace the variables $\mathbf{s}_1^*, \theta_1, t_1, \mathbf{s}_1'^*$ by their known values. It is guaranteed that the obtained expression is satisfiable. Its remaining quantifiers are eliminated yielding all the possible orientations of the second commanded velocity in the strategy. One value θ_2 is arbitrarily selected and the motion command $(\vec{\mathbf{v}}_{\theta_2}^c, \mathcal{Q}[\overline{\mathcal{F}}_{K_1,\mathbf{s}_2^*,\theta_2}(t) \cap \mathcal{U}_s(\mathbf{s}^*(t))] \subseteq \mathcal{G})$ is executed, with $\mathbf{s}_2^*$ being the pair of sensed configuration and force at time $t = 0$ (of the second motion). It is guaranteed to terminate in the goal $\mathcal{G}$.

Therefore, the above planning method forward chains from $\mathcal{I}$ in order to construct all the possible recognizable sets achievable from $\mathcal{I}$ and backward chains from $\mathcal{G}$ in order to determine which ones among these sets are preimages. The method gives a Tarski sentence that implicitly defines all the possible two-step strategies. Solving this sentence provides the commanded velocity of the first step and a sentence that defines the second commanded velocity. This sentence, which depends on parameters that are all known when the first motion terminates, allows the robot to select the second commanded velocity at execution time.

TESLA EXHIBIT 1013, Page 523

8.2.4 Multi-Step Strategy

Consider the problem of planning an r-step motion strategy to achieve $\mathcal{G}$ from $\mathcal{I}$. The above method extends in a straightforward fashion.

Let $K_k(\mathbf{s}_1^*, \theta_1, t_1, \mathbf{s}_1'^*, \mathbf{s}_2^*, \theta_2, ..., \theta_k, t_k, \mathbf{s}_k'^*)$, $k = 0, 1, ..., r$, denote a recognizable set after the execution of k motion commands. Let it be abbreviated to K_k. It is recursively defined as:

$$K_0 = \mathcal{I} \quad \text{and} \quad K_k = \mathcal{Q}[H_k], \quad \text{for } k = 1, ..., r,$$

with:

$$H_k = \overline{\mathcal{F}}_{K_{k-1}, \mathbf{s}_k^*, \theta_k}(t_k) \cap \mathcal{U}_s(\mathbf{s}_k'^*).$$

Now, let $P_k(\mathbf{s}_1^*, \theta_1, t_1, \mathbf{s}_1'^*, \mathbf{s}_2^*, \theta_2, ..., \theta_k, t_k, \mathbf{s}_k'^*)$ — we abbreviate it to P_k — be the condition which is satisfied when K_k is a preimage of $\mathcal{G}$ for a sequence of $r-k$ motion commands along the directions $\theta_{k+1}, ..., \theta_r$. P_k, $k = 0, ..., r-1$, is recursively defined as follows:

$$\begin{aligned} &P_k \equiv \\ &\exists \theta_{k+1} \in S^1,\ \forall \mathbf{s}_{k+1} \in (\widetilde{K_k})_{\theta_{k+1}},\ \forall \mathbf{s}_{k+1}^* \in \mathcal{U}_s(\mathbf{s}_{k+1}),\ \forall \tau_{k+1},\ \exists t_{k+1} \geq 0\ ,\ \forall \mathbf{s}_{k+1}'^* : \\ &\qquad (\tau_{k+1} \in \overline{\mathcal{D}}_{\theta_{k+1}}((\widetilde{K_k})_{\theta_{k+1}} \cap \mathcal{U}_s(\mathbf{s}_{k+1}^*)) \ \wedge\ \mathbf{s}_{k+1}'^* \in \mathcal{U}_s(\mathbf{s}_{\tau_{k+1}}(t_{k+1}))) \Rightarrow P_{k+1} \end{aligned}$$

with:

$$P_r \equiv [K_r \subseteq \mathcal{G}].$$

The sentence $\forall \mathbf{s}_0^* : P_0(\mathbf{s}_0^*)$ describes the existence of an r-step motion strategy. It is satisfiable if and only if there is one such strategy. If this is the case, a guaranteed strategy is executed by considering the sentences P_0 through P_{r-1} in sequence. P_k gives the guaranteed commanded velocity and the termination condition P_{k+1} for the $(k+1)^{\text{th}}$ motion command.

The sentence $\forall \mathbf{s}_0^* : P_0(\mathbf{s}_0^*)$ involves r existential quantifiers alternating with universal quantifiers. Hence, it requires double-exponential time in r to determine its satisfiability (using, say, Collins' method).

It is interesting to note that the existential quantifiers apply to the robot's motions, while the universal quantifiers apply to the actions of

TESLA EXHIBIT 1013, Page 524

nature (measured through the sensors). This alternation, which is classical in game theory, is investigated in [Taylor, Mason and Goldberg, 1987].

This alternation can be eliminated by removing sensing, for instance by using the sticking termination condition. Using this termination condition also results in using the timeless forward projection (defined as in Subsection 5.2) instead of the forward projection with time. Thus, we no longer need the somewhat arbitrary assumption that the effective commanded velocity takes at most p successive values during any actual motion. The method reduces to that presented in Subsection 5.2, which only requires time single-exponential in r.

The method can also be adapted to the following termination condition:

$$\mathcal{F}_\theta(\mathcal{P} \cap \mathcal{K}_\theta(\mathbf{q}^*(0), \vec{\mathbf{f}}^*(0))) \cap \mathcal{K}_\theta(\mathbf{q}^*(t), \vec{\mathbf{f}}^*(t)) \subseteq \mathcal{T}$$

where $\mathcal{F}_\theta(\mathcal{P})$ is the timeless forward projection of $\mathcal{P}$ for the commanded velocity $\vec{\mathbf{v}}_\theta^c$ and $\mathcal{K}_\theta(\mathbf{q}^*, \vec{\mathbf{f}}^*)$ is the set of configurations $\mathbf{q}$ θ-consistent with the sensed values $\mathbf{q}^*$ and $\vec{\mathbf{f}}^*$ (see Subsection 4.5). Like the sticking termination condition, this condition (which is slightly more powerful than the instantaneous sensing termination condition used in Subsection 4.5) does not require the effective commanded velocity to take p successive values. But it does involve sensing and yields a double-exponential time planner.

9 On Uncertainty in Model

Throughout this chapter, we have assumed that the geometry of $\mathcal{C}_{valid}$ was exactly known. In this last section, we briefly explore the effect of uncertainty in the geometry of $\mathcal{C}_{valid}$.

Consider the point-in-hole example shown in Figure 28. Assuming that all the angles of the C-obstacle are right angles and that the global orientation of the C-obstacle is perfectly known, the dimensions shown in Figure 28.a suffice to define the geometry of $\mathcal{C}_{valid}$. Consider the case where all these dimensions, except d (width of the hole), are exactly known. Assume further that $d \in (d_{min}, d_{max})$.

One way to represent this uncertainty is to extend the configuration space into a **generalized configuration space** by adding one dimen-

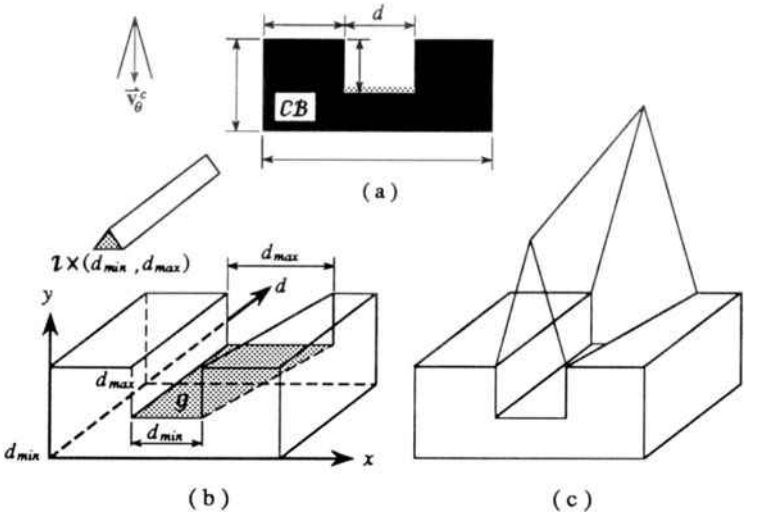


Figure 28. $\mathcal{C}_{valid}$ contains a single C-obstacle $\mathcal{CB}$ (Figure a). We assume that the geometry of $\mathcal{C}_{valid}$ is exactly known, except for the hole's width d, which is only known to be in the interval (d_{min}, d_{max}). This uncertainty is represented by adding a new dimension to the configuration space. We obtain the generalized configuration space shown in Figure b. The C-obstacle is now a three-dimensional region and $\mathcal{CB}$'s hole maps to a groove. The goal region $\mathcal{G}$ is the bottom face of the groove. A generalized preimage of this region for the commanded velocity $\vec{v}_\theta^c$ is shown in Figure c.

sion corresponding to the d parameter (see Figure 28.b) [Donald, 1987a]. Each cross-section of the generalized configuration space perpendicular to the d-axis is the configuration space for a particular value of d. In the generalized configuration space, the hole becomes a groove and the goal $\mathcal{G}$ is the bottom face of this groove. The initial region is the cylinder $\mathcal{I} \times (d_{min}, d_{max})$. A strongly guaranteed strategy should reach and terminate in $\mathcal{G}$. (Notice that, when this strategy is executed, the robot moves within a single cross-section at constant d. We do not know which one in advance. Even during execution, the robot will usually not collect sufficient information to determine in which cross-section it is moving.)

The concept of generalized configuration space extends to the case where there are several imprecisely known parameters in the description of $\mathcal{C}_{valid}$, by simply adding a new dimension for each such parameter. We

can define a preimage of a goal in the generalized configuration space as we did in a regular configuration space and the principle of the preimage backchaining approach still applies.

Assume as in the above example that there is a single parameter d, whose value is not known exactly. A preimage in the generalized configuration space — let us call it a *generalized preimage* — is a union of preimages computed at various values of d for the same commanded velocity (see Figure 28.c). Let us assume that we compute these preimages by backprojecting from the sticking edges in the goal. When the parameter d varies over its uncertainty interval, the topology of the preimage (computed at a specific value of d) changes at a finite number of critical values of d, and hence we can compute and represent the corresponding generalized preimage in much the same way as we did for nondirectional backprojections (Subsection 5.1.1). In the example of Figure 28, the only critical value of d is $d = 0$ (if $0 \in (d_{min}, d_{max})$), where the topology of $\mathcal{C}_{valid}$ changes. As another example, consider the case where the geometry of the C-obstacle region is exactly known, but its orientation is completely unknown (for instance, this could be the problem of meshing two planar gears [Donald, 1987a]). In this case, for any goal and any commanded velocity, the generalized preimage computed by backprojecting from the goal sticking edges is exactly the nondirectional preimage computed in Subsection 5.1.1 with the orientation of the C-obstacle region substituted for the orientation of the commanded velocity.

The planning method described in Subsection 8.2 is independent of the dimension of the planning space. Therefore, it also applies to motion planning with uncertainty in the input model, as long as this uncertainty involves a finite number of parameters [Canny, 1989].

Exercises

1: Justify the construction of the friction cone at a contact vertex, which is illustrated in Figure 3.

2: Assume perfect generalized damping control. $\mathcal{C}_{valid}$ is defined as in Section 1. Let $\mathbf{q}_s$ be the starting configuration of two motions commanded along two different velocities $\vec{\mathbf{v}}^c_{\theta_1}$ and $\vec{\mathbf{v}}^c_{\theta_2}$. Let τ_1 and τ_2 be the respective trajectories followed by these motions. Show that τ_1 and τ_2

TESLA EXHIBIT 1013, Page 527

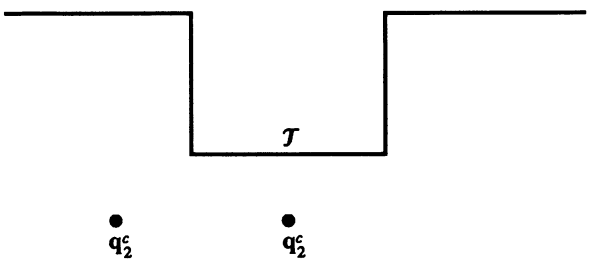


Figure 29. The configurations $\mathbf{q}_1^c$ and $\mathbf{q}_2^c$ represents two desired configurations given to a robot with generalized spring control (Exercise 4).

have no loops, and that they can meet (and possibly separate again), but never cross each other [Friedman, Hershberger and Snoeyink, 1990].

3: Consider the point-in-hole example of Figure 9. Assume non-zero friction in contact space. What is the union of all the preimages of the goal $\mathcal{T}$ when the orientation of the commanded velocity varies over S^1?

4: Generalized damping control is only one possible model of force-compliant motion control. Another interesting model is *generalized spring* control [Salisbury, 1980]. In this model, the control statement is parameterized by a desired configuration $\mathbf{q}^c$ and the robot's commanded trajectory τ^c is defined by the following equation:

$$\vec{\mathbf{f}}(t) = B\vec{\mathbf{v}}^c(t) - K(\mathbf{q}^c - \tau^c(t))$$

where B and K are 2×2 matrices and $\vec{\mathbf{v}}^c(t) = d\vec{\tau}^c(t)/dt$ is the velocity of the robot at time t along the commanded trajectory.

Assume here that B and K are constant diagonal matrices, each with equal positive diagonal coefficients, and that $B = K$. In free space, $\vec{\mathbf{f}}(t) = 0$, so that $\vec{\mathbf{v}}^c(t) = \mathbf{q}^c - \tau^c(t)$; hence, the robot moves along a straight trajectory pointing toward $\mathbf{q}^c$. In contact space, if the vector $-(\mathbf{q}^c - \tau^c(t))$ points into the friction cone at $\tau^c(t)$, then the robot sticks, otherwise it slides.

Consider the point-in-hole example shown in Figure 29. Assume perfect generalized spring control and non-zero friction. Construct the preimage of the bottom edge $\mathcal{T}$ for each of the two desired configurations $\mathbf{q}_1^c$ and

$\mathbf{q}_2^c$ shown in the figure. (Assume perfect position sensing so that in both cases the termination condition is $\mathbf{q}^*(t) \in \mathcal{T}$.)

5: Give an example of a planning problem which cannot be solved using the preimage backchaining approach, although the goal is achievable by executing a finite (but unbounded) number of motion commands [Mason, 1984].

6: A *weak preimage* of a goal for a motion command $\mathrm{M} = (\vec{\mathbf{v}}_\theta^c, \mathtt{TC})$ is any region $\mathcal{P} \subseteq \mathcal{C}_{valid}$ such that a motion commanded along $\vec{\mathbf{v}}_\theta^c$ and starting from within $\mathcal{P}$ may fortuitously attain the goal and terminate in the goal. Consider the point-in-hole of Figure 9. Construct the maximal weak preimage of the goal $\mathcal{T}$. Consider any initial region $\mathcal{I}$ contained in this weak preimage. Construct a one-step weakly guaranteed strategy (see the remark at the end of Subsection 1.4) that either achieves $\mathcal{T}$ from $\mathcal{I}$ or fails recognizably.

7: Show the following two properties:

- If $\mathcal{P}$ is a preimage of $\mathcal{T}$ for M, then any subset of $\mathcal{P}$ is also a preimage of $\mathcal{T}$ for M.
- If $\mathcal{P}_1$ and $\mathcal{P}_2$ are both preimages of $\mathcal{T}$ for M, then $\mathcal{P}_1 \cup \mathcal{P}_2$ is also a preimage of $\mathcal{T}$ for M.

8: Implement a simple preimage backchaining motion planner using the backprojection from sticking edges for computing preimages and Erdmann's algorithm for computing backprojections. Select the commanded velocities from a predefined finite set.

9: Consider the three goals $\mathcal{T}_1$, $\mathcal{T}_2$ and $\mathcal{T}_3$ shown in Figure 30. The commanded velocity $\vec{\mathbf{v}}_\theta^c$ is pointing vertically downward. The position uncertainty disc is as shown in the figure. Assume perfect force sensing. Construct the θ-kernel of each of the three goals.

10: Consider the goal region $\mathcal{T}$ in Figure 17.a. Construct all the θ-kernels of $\mathcal{T}$ when θ varies over S^1.

11: Propose an algorithm for computing the θ-kernel of a goal described as a polygon in $\mathcal{C}_{valid}$.

TESLA EXHIBIT 1013, Page 529

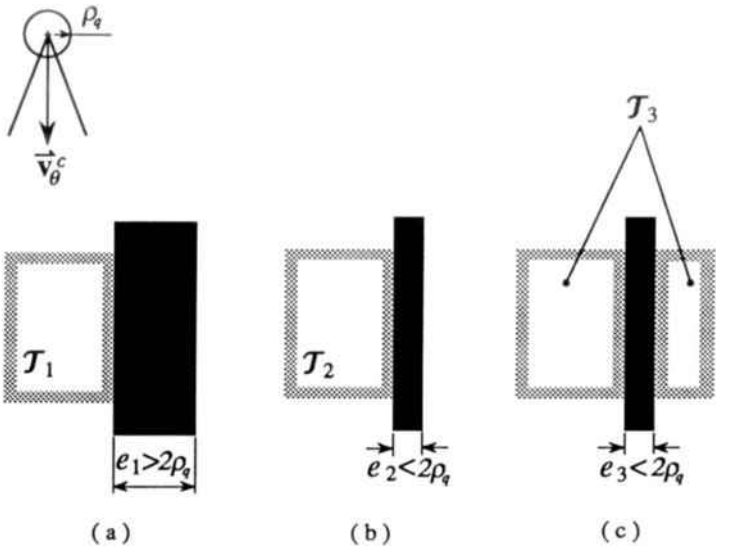


Figure 30. Each of the goal regions, $\mathcal{T}_1$, $\mathcal{T}_2$ and $\mathcal{T}_3$, consists of one or two polygons in valid space. The radius of the position uncertainty disc is ρ_q. The commanded velocity is pointing vertically downward (Exercise 9).



Figure 31. The robot moves in a horizontal line. The initial region is $\mathcal{I}$ and the goal region is $\mathcal{G}$. The goal region is a closed segment whose length is exactly $2\rho_q$, which is the same as for the open interval representing the uncertainty in position sensing. The goal can be achieved recognizably using a termination condition with final state (Exercise 13).

12: Consider Figure 23. Show the regions F_1, F_2 and F_3 defined as in Subsection 7.2.

13: Assume that the robot's configuration space is a horizontal line (i.e. $\mathcal{C} = \mathbf{R}$). There is no C-obstacle (i.e. $\mathcal{C}_{valid} = \mathbf{R}$). The uncertainty in position sensing is an open segment of length $2\rho_q$ centered at the

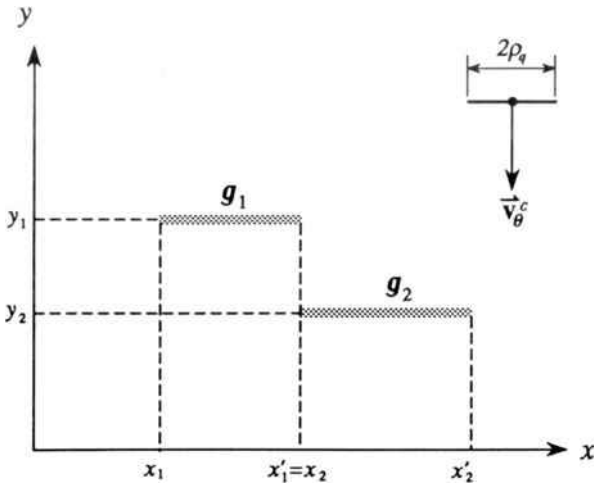


Figure 32. The goal consists of two horizontal segments in a configuration space with no C-obstacle (Exercise 14).

measured configuration. There are two possible velocities: "rightward" and "leftward". The goal region $\mathcal{G}$ is a closed segment of length $2\rho_q$ as shown in Figure 31. The initial region $\mathcal{I}$ is a segment of finite length on the left of $\mathcal{G}$. The selected commanded velocity is "rightward". Construct a computable form of the termination condition with final state that is guaranteed to terminate the motion in $\mathcal{G}$. What are the preimages of $\mathcal{G}$ for this termination condition?

14: Let $\mathcal{C} = \mathcal{C}_{valid} = \mathbf{R}^2$ (there is no obstacle). Assume perfect robot control and perfect position sensing in the vertical direction. Hence, $\mathcal{U}_q(\mathbf{q}^*)$ is an open horizontal segment of length $2\rho_q$ centered at the measured configuration $\mathbf{q}^*$. The goal region is $\mathcal{G} = \mathcal{G}_1 \cup \mathcal{G}_2$ (see Figure 32), where:

- $\mathcal{G}_1$ is a horizontal line segment connecting (x_1, y_1) to (x_1', y_1),
- $\mathcal{G}_2$ is a horizontal line segment connecting (x_2, y_2) to (x_2', y_2), with $x_2 = x_1'$.

The commanded velocity is pointing vertically downward.

Construct a preimage of $\mathcal{G}$ that is maximal over all possible termination conditions and specify the corresponding termination condition. Is this maximal preimage unique? Characterize all the preimages that are maximal over all possible termination conditions [Erdmann, 1984].

15: Consider the Tarski sentence of Subsection 8.2.3 which expresses the existence of a two-step strategy to achieve a goal $\mathcal{G}$. Rewrite this sentence by using the termination condition:

$$\mathcal{F}_\theta(\mathcal{P} \cap \mathcal{K}_\theta(\mathbf{q}^*(0), \vec{\mathbf{f}}^*(0))) \cap \mathcal{K}_\theta(\mathbf{q}^*(t), \vec{\mathbf{f}}^*(t)) \subseteq \mathcal{T}$$

where $\mathcal{F}_\theta(\mathcal{P})$ is the timeless forward projection defined as in Subsection 5.2 and $\mathcal{K}_\theta(\mathbf{q}^*, \vec{\mathbf{f}}^*)$ is the set of configurations $\mathbf{q}$ θ-consistent with the sensed values $\mathbf{q}^*$ and $\vec{\mathbf{f}}^*$ (see Subsection 4.5). Show that the new sentence remains a Tarski sentence even if we drop the condition that the effective commanded velocity takes a fixed number of successive values during any actual motion.

16: Consider an omnidirectional mobile robot equipped with odometric and proximity range sensors. Discuss the application of the preimage backchaining approach to the generation of motion strategies for this robot.

17: An approach to motion planning with uncertainty, known as *skeleton refinement*, has been presented in [Taylor, 1976], [Lozano-Pérez, 1976], [Brooks, 1982] and [Pertin-Troccaz and Puget, 1987]. Compare this approach to preimage backchaining.

TESLA EXHIBIT 1013, Page 532