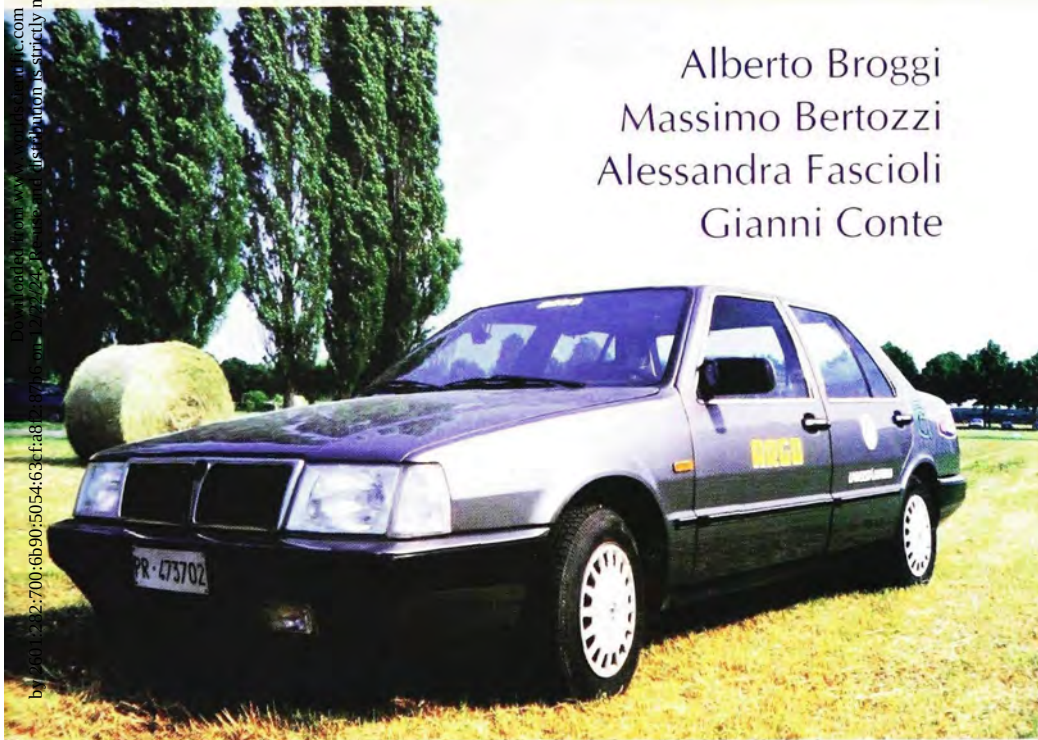


Downloaded from www.worldscientific.com by 203.122.22.700 on 05/05/24. This paper is user-generated and its use is strictly not permitted, except for Open Access articles.

Automatic Vehicle Guidance: The Experience of the ARGO Autonomous Vehicle

Alberto Broggi
Massimo Bertozzi
Alessandra Fascioli
Gianni Conte



World Scientific

Automatic Vehicle Guidance: The Experience of the ARGO Autonomous Vehicle

This page is intentionally left blank

Automatic Vehicle Guidance: The Experience of the ARGO Autonomous Vehicle

Alberto Broggi
Massimo Bertozzi
Alessandra Fascioli
Gianni Conte

University of Parma, Italy



World Scientific

Singapore • New Jersey • London • Hong Kong

Published by

World Scientific Publishing Co. Pte. Ltd.

P O Box 128, Farrer Road, Singapore 912805

USA office: Suite 1B, 1060 Main Street, River Edge, NJ 07661

UK office: 57 Shelton Street, Covent Garden, London WC2H 9HE

British Library Cataloguing-in-Publication Data

A catalogue record for this book is available from the British Library.

AUTOMATIC VEHICLE GUIDANCE:

The Experience of the ARGO Autonomous Vehicle

Copyright © 1999 by World Scientific Publishing Co. Pte. Ltd.

All rights reserved. This book, or parts thereof, may not be reproduced in any form or by any means, electronic or mechanical, including photocopying, recording or any information storage and retrieval system now known or to be invented, without written permission from the Publisher.

For photocopying of material in this volume, please pay a copying fee through the Copyright Clearance Center, Inc., 222 Rosewood Drive, Danvers, MA 01923, USA. In this case permission to photocopy is not required from the publisher.

ISBN 981-02-3720-0

This book is printed on acid-free paper.

Printed in Singapore by Uto-Print

*"... and at night, by the skill of Argo,
they reached broad-flowing Phasis,
and the utmost bourne of the sea."*

Apollonius, THE ARGONAUTICA, Book II

This page is intentionally left blank

Preface

During the last decade, the subject of Intelligent Transportation gained strategic importance and widespread relevance. Many projects were launched worldwide aimed at analyzing the problem of people's mobility and goods transportation from a number of different perspectives; and it is in the last few years that the first prototypes of both vehicles equipped with automatic driving facilities and road infrastructures supporting these functionalities, are being tested and demonstrated to the public. This book surveys the history of intelligent vehicles, discusses some of the different approaches developed worldwide by a large number of research institutions, and presents the solutions adopted by the *Università di Parma* in the ARGO project, which started about 10 years ago within the Eureka PROMETHEUS project. In particular, this book illustrates the problem, proposes some of the different solutions, and details the design, the development, and the engineering of a hardware and software platform for Automatic Vehicle Guidance, as well as the set-up of two prototype vehicles.

Among the main results of this research, the GOLD (Generic Obstacle and Lane Detection) system is presented; it is an automatic driving system which has been integrated on ARGO –a Lancia Thema 2000 passengers' car– allowing to drive the vehicle autonomously in real traffic conditions along highways and freeways, with no requirements of additional specific road infrastructures.

The results of this long term research was demonstrated to the international scientific community and to the public in the first week of June 1998 with a journey through Italy, the *MilleMiglia in Automatico* tour, during

which the vehicle drove autonomously for about 2000 km. The experience of this demonstration is discussed in the book, along with a description of the main advantages and problems encountered.

This book is divided into three parts. The first part presents the motivation of this research and a brief history of the main projects launched worldwide aimed at vision-based vehicle driving. The second and third parts are related to the ARGO project. Part II describes both the algorithms and hardware platforms developed during the whole project –starting from the very first implementation, up to the current one– and presents the equipment installed on the ARGO prototype vehicle. Part III shows a report of the extensive test that was performed on ARGO and analyzes the problems encountered and the overall system performance.

Acknowledgment

During the last ten years of research a large number of people helped and supported this project. Among them we would like to acknowledge: Francesco Gregoretti, Leonardo Reyneri, Claudio Sansoè, and Roberto Passerone from the *Politecnico di Torino*, Italy, for the joint development of the PAPRICA hardware system; Giorgio Quaglia and Sandra Denasi from the *Istituto Elettrotecnico Nazionale Galileo Ferraris*, Torino, Italy, and Giovanni Adorni from our department for the set-up of MOB-LAB; Lucio Bianco, Leopoldo Chinaglia, and Agostino Scognamiglio from *Progetto Finalizzato Trasporti II*, CNR, Italy, for supporting the research; Sergio Temperi and Edmondo Pietranera from Telecom Italia Mobile, which sponsored the *MilleMiglia in Automatico* tour and all our friends who welcomed us as guests, hosted our seminars and helped in the organization of this exciting journey; Mrs. Lucia Caccioli for her kind help in improving the style of the presentation; and finally our administrative and technical staff whose contribution was essential for the complete development of the whole project. We would also like to acknowledge the support we received from Giorgio Quaglia, Giulio Vivo, Uwe Franke, Wilfried Enkelmann, Pierre Charbonnier, Dean Pomerleau, Ümit Özgüner, Keith Redmill, Ichiro Masaki, and Nicole Love, who provided up-to-date material regarding the state of the art of their projects.

This page is intentionally left blank

Contents

Preface	vii
Acknowledgment	ix
PART I INTELLIGENT VEHICLES	1
1 Introduction	5
2 Intelligent Vehicles and Machine Vision	9
2.1 Evolution of Intelligent Transportation Systems	9
2.2 Requirements of Intelligent Transportation Systems	12
2.3 Sensing the Environment	14
2.4 Machine Vision	15
3 State of the Art	21
3.1 Road Following	21
3.1.1 Lane Detection	21
3.1.2 Obstacle Detection	23
3.2 Worldwide Projects	25
3.2.1 Research carried out on the MOB-LAB Vehicle	25
3.2.2 Research carried out at the <i>Centro Ricerche FIAT</i>	29
3.2.3 Research carried out at the <i>Universität der Bundeswehr</i>	34
3.2.4 Research carried out at <i>Daimler-Benz</i>	37
3.2.5 Research carried out at the <i>Fraunhofer-Institut für In-</i> <i>formations und Datenverarbeitung</i>	38

3.2.6	Research carried out at the <i>Laboratoire Central des Ponts-et-Chaussées de Strasbourg</i>	40
3.2.7	Research carried out at the <i>Defence Evaluation & Research Agency</i>	42
3.2.8	Research carried out at <i>Carnegie Mellon University</i>	44
3.2.9	Research carried out at <i>The Ohio State University</i>	47
3.2.10	Research carried out at the <i>University of Michigan</i>	49
3.2.11	Research carried out at the <i>Massachusetts Institute of Technology</i>	51
3.2.12	Research carried out at the <i>Phoang University of Science and Technology</i>	52
PART II	THE ARGO PROJECT	55
4	Algorithms for Image Processing	59
4.1	Lane Detection: a Model-Based Approach	59
4.1.1	The multi-resolution approach	60
4.1.2	The algorithm structure	61
4.1.3	Performance analysis	69
4.1.4	Critical analysis and evolution	70
4.2	Obstacle Detection: a Model-Based Approach	74
4.2.1	The vehicle detection algorithm	75
4.2.2	Performance analysis	80
4.3	The GOLD System	82
4.3.1	Inverse Perspective Mapping	83
4.3.2	Inverse Perspective Mapping and stereo vision	88
4.3.3	Functionalities	94
4.3.4	An extension of Inverse Perspective Mapping to handle non-flat roads	107
4.3.5	Discussion	114
5	Hardware Support for Real-Time Image Processing	123
5.1	The PAPRICA Architecture	123
5.1.1	Architectural issues	124
5.1.2	Hardware system description	125
5.2	Critical Analysis of the PAPRICA Architecture	128
5.2.1	Memory organization and processor virtualization	128
5.2.2	I/O problems	129

5.2.3	Instruction set	130
5.2.4	Architectural evolution	131
5.3	The PAPRICA-3 Architecture	132
5.3.1	Hardware system description	133
5.3.2	Obstacle Detection on PAPRICA-3	137
5.4	The MMX Technology	142
5.4.1	MMX optimization issues	143
5.4.2	Obstacle Detection on an MMX-based processor	146
5.5	Comparison between PAPRICA-3 and MMX Processors	152
5.5.1	Algorithms implementation	152
5.5.2	Performance evaluation	155
5.5.3	Discussion	156
6	The ARGO Vehicle	161
6.1	Description	161
6.1.1	The name	162
6.2	The Data Acquisition System	163
6.2.1	The vision system	163
6.2.2	The speed sensor	165
6.2.3	The user interface	165
6.2.4	The keyboard	166
6.3	The Processing System	166
6.4	The Output System	167
6.4.1	The acoustical devices	168
6.4.2	The optical devices	168
6.4.3	The mechanical devices	169
6.5	The Control System	171
6.5.1	Lane change maneuvers	171
6.6	Functionalities	173
6.7	Other Vehicle Equipments and Emergency Features	174
PART III	THE PROJECT'S RESULTS	177
7	The <i>MilleMiglia</i> in Automatico Tour	181
7.1	Description	181
7.1.1	Dates and schedule	182
7.1.2	Data logging	183
7.1.3	Live broadcasting of the event via Internet	183

8 Performance Analysis	187
8.1 System Performance	187
8.1.1 The vision system	187
8.1.2 The processing system	188
8.1.3 The visual processing	188
8.1.4 The control system	189
8.1.5 The man-machine interface	190
8.1.6 Environmental conditions	190
8.1.7 The data transmission system	192
8.2 Statistical Analysis of the Tour	193
8.2.1 Detailed analysis of one hour of automatic driving . . .	194
8.3 Discussion	197
9 Closing Remarks	201
APPENDICES	203
A Morphological Implementation of the DBS Filter	205
B PAPRICA-3 Programming Environment	211
B.1 Low Level Programming Language	211
B.2 High Level Programming Language	213
B.3 Assembly Code Optimization	214
B.3.1 Deterministic optimization	214
B.3.2 Stochastic optimization	214
B.3.3 Parallel implementation of the code optimizer	218
C Global Communications on PAPRICA-3	219
C.1 Concurrent Communications on the ICN	221
C.2 Determining the Sets of Compatible Communications	225
References	227
Biographic Notes	241

PART I

INTELLIGENT VEHICLES

Downloaded from www.worldscientific.com
by 2601:282:700:6b90:5054:63cf:a8f2:87b6 on 12/22/24. Re-use and distribution is strictly not permitted, except for Open Access articles.

This page is intentionally left blank

This first part of the book presents the motivations that triggered off the high interest towards Intelligent Transportation Systems and describes the social and economical impact that automatically driven vehicles will cause in the future. Then, after the analysis of the main requirements and consequential benefits, a brief history of Intelligent Transportation Systems is presented. Finally, it describes the results obtained in this field by various research institutions worldwide.

This page is intentionally left blank

Chapter 1

Introduction

Automatic Vehicle Driving is a generic term used to address a technique aimed at automating –entirely or in part– one or more driving tasks. The automation of these tasks carries a large number of benefits, such as: a higher exploitation of the road network, lower fuel and energy consumption, and –of course– improved safety conditions compared to the current scenario.

The tasks that automatically driven vehicles are able to perform range from the following: the possibility to follow the road and keep within the right lane, maintaining a safe distance between vehicles, regulating the vehicle's speed according to traffic conditions and road characteristics, moving across lanes in order to overtake vehicles and avoid obstacles, helping to find the correct and shortest route to a destination, and the movement and parking within urban environments.

Besides the obvious advantages linked to increasing safety and reducing road accidents –thus saving human lives– the possibility of having vehicles riding in a much closer proximity than they do today, produces an increment of road capacity, and, together with the intelligent modulation of vehicles' speed, also causes an appreciable reduction of fuel consumption. In other words, Automatic Vehicle Driving tends to achieve optimal use of current infrastructures, improve mobility, and minimize risks, travel times, and energy consumption. Moreover, commercial and industrial vehicles which repeatedly move along a given path, benefit from a stronger control of their routes and require less personnel to manage their moves. Co-operative driving and dynamic fleet management can therefore play a fundamental role for the reduction of industrial costs.

Two main cooperative solutions are possible to achieve automatic driving functionalities: they require to act on

- infrastructures or
- vehicles.

Both scenarios have their own pros and cons, depending on the specific application. As a general rule, public transportation and industrial robotics applications –based on repetitive and prescheduled routes– can generally benefit from a specific and fixed road infrastructure. Instead, for private vehicles, moving along extremely large and extended road networks, the organization, set-up, and maintenance of nation-wide road infrastructures becomes cumbersome and extremely expensive.

Conversely, the establishment of a structured environment specifically designed for this purpose can be possibly considered for a reduced subset of the road network, for example for building a fully automated highway on which only automatic vehicles –public or private– can drive.

Moreover, infrastructure-based systems generally require a more extensive preparation time. For this reason, systems that are expected to be achieved on a short-term basis can only be vehicle-autonomous or at the most, simple versions of infrastructure-based systems. Such systems do not require a dedicated complex infrastructure or fit into already existing traffic infrastructure (RDS, GSM, beacons for automatic toll systems and traffic data banks).

Furthermore, Automatic Vehicle Driving hides another extremely important benefit, that first triggered the interest in this new field of application. The on-board equipment and technology that allows a vehicle to drive autonomously can also be exploited as a safety enhancement in a more general meaning. The autopilot can be switched to the more static task of monitoring the driver's activities. In case of dangerous situations –such as driver's illness or drowsiness–, the equipment understands the risk and warns the driver or even takes control of the vehicle, starting an emergency maneuver and stopping the vehicle on the emergency lane, or keeping the vehicle on the driving lane at a constant speed until the driver resumes control. The autopilot, considered in this second function, behaves like an *active* security system able to prevent accidents, like the Anti Breaking System, ABS.

Recently a large emphasis has been given to security systems –both active and passive ones– and almost every automotive company is integrating

such systems into its latest car models. For this reason, the techniques and the technologies supporting Automatic Vehicle Driving are currently being studied by various parties worldwide, focusing on different perspectives. Automobile manufacturers view automatic driving systems as an interesting product; public and government institutions, such as public transportation companies, mainly focus on the possibility of increasing safety, reducing energy consumption, and optimizing comfort and traffic control. Similarly these kind of activities have also been tackled by the academic world which, together with military institutions, sees the problem of Automatic Vehicle Guidance as a process of technological transfer from avionics, underwater, military, or robotics applications in general, and as an engineering problem to produce low-cost systems for the private market.

In the last decade, a large number of research institutes worldwide have been involved in national and international projects related to the analysis of the *Intelligent Transportation Systems* problem, and a number of prototypes of intelligent vehicles have been designed, implemented, and tested on the road.

The design of these prototypes was preceded by the analysis of solutions deriving from similar and close fields of research, and exploded with a great flourishing of new ideas, innovative approaches, and novel ad-hoc solutions. robotics, artificial intelligence, computer science, computer architecture, telecommunications, control and automation, signal processing are just some of the principal research areas from which the main ideas and solutions were first derived. Initially, underlying technological devices—such as head-up displays, infrared cameras, radars, sonars—derived from expensive military applications, but, thanks to the increased interest in these applications and progress of industrial production, today's technology offers sensors, processing systems, and output devices at very competitive prices. In order to test a wide spectrum of diverse approaches, these prototypes of automatic vehicles are equipped with a large number of different sensors.

These vehicles are currently undergoing extensive tests both on structured environments, such as closed roads or private tracks, and on public roads in real traffic conditions. The first preliminary results of these tests demonstrated that full automation of traffic (at least on motorways or sufficiently structured roads) is technically feasible. Nevertheless, not all problems related to automatic vehicle driving that still remain to be solved are of technical nature: there are some aspects that must be considered

and carefully evaluated in the design of such systems. First of all, prior to having an automatic driving system sold and installed on a commercial vehicle, all the legal aspects related to the responsibility in case of faults and incorrect behavior of the system must be solved. Secondly, in case no specific roads are built and dedicated to automatic vehicles only, the possibility of driving on a motorway along with automatic vehicles must be considered and its impact on human drivers evaluated. Although technical aspects seem to have a higher importance, these problems must be dealt with and solved as well, since they represent the basics and prerequisites on which future automatic highways will rely.

The research carried out on intelligent vehicles –after its first explorative stage– has now reached its second stage: many different approaches have been evaluated during these years and the most promising ones are being engineered, installed on prototype vehicles, and extensively tested on the road. However, a long period of exhaustive tests and refinement must precede the availability of these systems on the general market, and a fully automated highway system with intelligent vehicles driving and exchanging information is not expected for a couple of decades. For the time being, complete automation will be restricted to special infrastructures such as industrial applications or public transportation. Then, automatic vehicle technology will be gradually extended to other key transportation areas such as the transportation of goods, for example on expensive trucks, where the cost of an autopilot is negligible with respect to the cost of the vehicle itself and the service it provides. Finally, once technology has been stabilized and after freezing the most promising solution and the best algorithms, a massive integration and a widespread use of such systems will also take place with private vehicles, but this will not happen for another two or more decades.

Chapter 2

Intelligent Vehicles and Machine Vision

2.1 Evolution of Intelligent Transportation Systems

After completion of the first explorative phase, that started around the beginning of the eighties, the field of Intelligent Transportation Systems (ITS) is now entering its second phase characterized by a maturity in its approaches and by new technological possibilities which allow the development of the first test products. Nevertheless, the evolution towards the third stage, i.e. the massive production and integration of ITS technology on commercial vehicles, is expected to begin not before a couple of decades.

The interest in ITS technologies and their related issues was born about 20 years ago, when the problem of people and goods mobility began to arise: the saturation of the most common way to increase mobility, namely the extension of the road network, focused the interest of governments and research institutions towards new alternative solutions.

The main target of this first phase was to create a technical substrate to be used in the following prototyping stages, namely a formal basis covering different subjects. For this reason, automatic vehicle driving, intelligent route planning, and other extremely high-level functionalities were selected as main goals.

Government institutions activated the initial explorative phase by means of different projects worldwide. The first results were a deep analysis of the problem and the development of a feasibility study to understand the requirements and possible effects of the application of ITS technology. One of the main advantages of these activities was the possibility to have a large number of different and complementary research units –with differ-

ent backgrounds— working in a cooperative way. This research stage, in fact, mainly characterized by non-competitive research, produced a large number of different prototypes and possible solutions, all based on rather different approaches.

In this book, only in-vehicle applications will be considered and discussed, while road infrastructure, inter-vehicle communication, satellite communication, and route planning issues will not be covered.

In Europe the PROMETHEUS project (PROgram for a European Traffic with highest Efficiency and Unprecedented Safety) started this explorative stage in 1986. The project involved more than thirteen vehicle manufacturers and a large number of research units from governments and universities of 19 European countries. Within this framework, a number of different approaches regarding ITS were conceived, implemented, and demonstrated. The vehicle prototypes developed during this period were demonstrated at the end of the project in Paris, in October 1994. The most interesting approaches that were proposed included the VaMoRs test vehicle developed by the group of *Universität der Bundeswehr* [67]. It was able to follow the lane thanks to a pair of forward looking cameras and a custom parallel image processing engine. *Daimler-Benz* proposed a system called VITA [152] equipped with more than a dozen cameras and a large number of Transputer nodes and custom DSPs; it was demonstrated to move autonomously on highways at speeds up to 110 km/h.

In the United States a great deal of initiatives were launched to deal with the problems of mobility, involving many universities and research centers. Amongst them the *Carnegie Mellon University* research group, active for years in this field, developed the first NavLab vehicle [148] in 1985. After this, a large number of different approaches were tested on the series of NavLab vehicles, all developed by *CMU*; these included: ALVINN [80], a 30×32 neural net able to learn from a driver and to drive the vehicle thanks to the analysis of low-resolution visual patterns, and SCARF [40], a system based on color segmentation of pairs of images able to detect even unstructured roads in slow varying conditions of light. *General Motors* also presented a system for car following, based on custom hardware.

At the conclusion of this explorative phase—in 1995—the US established the National Automated Highway System Consortium, including many different institutions amongst which is the *University of California* and *Carnegie Mellon University*, who took part in an impressive demon-

stration in San Diego in August 1997 [13].

In the same way, also in Japan, where the mobility problem is much more intense and evident, some vehicle prototypes were developed within the framework of different projects. Similarly to what happened in the US, in 1996 the Advanced Cruise-Assist Highway System Research Association (AHSRA) was established amongst a large number of automobile industries and research centers, which demonstrated different approaches to the problem of Automatic Vehicle Guidance [150].

One of the aspects shared by all these first prototypes developed within the first research phase is the massive use of custom hardware. This recurrent choice was motivated by the fact that during those years (at the end of the eighties or beginning of the nineties) the hardware available on the market at a reasonably low cost was not powerful enough to provide real-time processing. This problem became even more intense, when the processing of images was involved. Therefore, besides selecting the proper sensors and developing specific algorithms, a large percentage of this first research stage was dedicated to the design, implementation, and test of new ad-hoc hardware platforms. The processing engines used to speed-up image processing tasks were mainly based on either SIMD computational paradigms (namely a substantial number of simple processors performing the same operation on a large set of data, usually image pixels) exploiting the spatial parallelism of images, or MIMD ones where the processors perform different functions on data subsets, therefore exploiting the parallelism of the algorithms. It is essential to emphasize that when a new computer architecture is built, not only the hardware aspects –such as instruction set, I/O interconnections, or computational paradigm– need to be considered, but software issues as well. Low-level basic libraries must be developed and tested along with specific tools for code generation, optimization and debugging.

Thanks to the improvement of technology, the situation has changed in the last few years, and almost all research groups are shifting towards the use of off-the-shelf components for their systems, since nowadays commercial hardware has reached a low price/performance ratio. Nevertheless, although the new generation systems are all based on commercial hardware, the development of custom hardware has not lost significance, but is gaining a renewed interest for the production of embedded systems. Once a hardware and software prototype has been built and extensively tested, its

functionalities must be frozen and integrated in a fully optimized and engineered embedded system before marketing. It is in this stage of the project that the development of ad-hoc custom hardware still plays a fundamental role and its costs are justified through a large scale market.

2.2 Requirements of Intelligent Transportation Systems

Any on-board system for ITS applications that is to be sold on the market needs to meet some important requirements which are summarized in the following.

Robustness

The final system that will be installed on a commercial vehicle needs to be robust with respect to a number of different aspects. First of all, any automatic system –even a very simple one such as an intelligent cruise control, which controls only the vehicle's speed– must be smart enough to adapt to different conditions of environment (flat or hilly paths), road (highways, extra-urban, or urban scenarios), traffic (different volumes of vehicles and obstacles), illumination (day, night, sunset, sunrise), and weather (sun, fog, rain, or even snow), and to tolerate unexpected changes of their parameters. The system must minimize –or better still reduce to zero– the number of incorrect results signals since they may affect the user's confidence over the system, and induce the driver to take over the commands of the automatic system and/or switch it off. Moreover, not only does the algorithms' robustness have to be granted, but also the hardware system –sensors included– requires the resistance to mechanical and thermic stress, such as vibrations or high temperatures. In other words, the whole system must be *automotive-engineered*.

Reliability

On-board systems for ITS applications are safety critical systems for which a high degree of reliability is required. This entails that the project has to be thorough and rigorous during all its phases, from the requirements specification to the design and implementation. An extensive phase of testing and validation is therefore of paramount importance.

Production and Operative Costs

It has been estimated that for marketing reasons an ITS system should cost no more than 10% of the vehicle price. These cost issues present no problems for trucks, buses, or high-cost vehicles in general, for which the installation of an expensive ITS system can be regarded as an investment on the vehicle's security and safety. Conversely, for the car market the cost issues become fundamentally important, requiring a specific engineering phase which may also need the redesign of the complete hardware platform. However, the development costs due to the system re-engineering will be compensated by the large car market.

Furthermore, not only do production costs of ITS apparatus need to be carefully considered and reduced as much as possible, but the operative costs need to be kept low as well. Power consumption is in fact another key parameter that must be considered for the car market, since the vehicle's performance should not be altered by the use of ITS apparatus.

Size and Design

The requirements of the car market are very specific: not only must the vehicle's performance not be altered and ITS production costs kept low, but also car styling should not suffer from the installation of new hardware and—in particular—new sensors for ITS applications. Therefore these systems need to be compact in size and the sensors need to be installed in a position that does not cause disturbance. Although for many sensors their positioning does not present aesthetical problems (for example, radars are typically positioned on the vehicle's front bumpers), the installation of cameras may be considered as an additional problem: some approaches require the cameras to be installed inside the driving cabin behind the windscreen in a high position.

Usability

Since ITS systems must be triggered off and controlled by a human driver, they need a user-friendly man-machine interface allowing the driver to switch on the system, resume control, and modify driving parameters on the fly, with just a few keystrokes. These parameters can be: speed, route, smoothness or alertness in driving. The new generation of steering wheels already incorporate commands for cellular phones, radio tuning, or cruise control; they

are also expected to become the front-end for ITS systems too. Furthermore, the use of visual signals (leds, lights, control panels, or even on-board monitors), mechanical feedbacks (steering wheel, seat, or pedal vibrations), and vocal messages used as driving assistance warnings must all be carefully evaluated. The system—especially a driving assistance one—must not flood the driver with a large quantity of information since this could reduce the driver's attention and cause nothing else but problems.

2.3 Sensing the Environment

The main requirement of ITS systems is a way of sensing the surrounding environment. Five different ways to perceive information about the environment commonly used in indoor robotics are tactile, acoustic, laser, radar, and vision sensors.

Tactile sensors

Tactile sensors are generally used in indoor robotics, where the speeds of robots are sufficiently low so as not to cause serious damages when hitting obstacles. Obviously in automotive applications tactile sensors, such as bumpers, are of no practical use.

Acoustic sensors

Acoustic sensors are interesting mainly because of their low cost, but, due to their very reduced detection range, are seldom used in automotive applications. These kind of sensors are classified as *active sensors*, because their use involves the measurement of alterations of signals emitted by the sensors themselves.

Laser-based sensors

Based on the Doppler effect, laser-based radars detect the distance of objects by measuring the travel time of a signal emitted by the sensors themselves and reflected by the object. Therefore, also laser-based sensors are *active sensors*. Their principal drawbacks are the low spatial resolution and slow scanning speed.

Radar-based sensors

Millimeter-wave radars share the same principle of laser-based radars, but are more expensive than laser-based radars. Conversely, millimeter-based radars are more robust to rain and fog than laser-

based radars. Unfortunately, both of them also share the same problems of low spatial resolution and slow scanning speeds. Again, radar-based sensors are classified as *active sensors*.

Vision-based sensors

Vision-based sensors are *passive sensors* and their use has several advantages over radar, laser, acoustic, and tactile sensors: for example the possibility to acquire data in a non-invasive way, thus not altering the environment.

In this case the scanning of the image is performed fast enough for ITS applications. Moreover, they can be used for some specific applications for which visual information plays a basic role (such as: lane markings localization, traffic signs recognition, obstacle identification) without any modifications to road infrastructures. Unfortunately vision sensors are less robust than millimeter-wave radars in foggy, night, or direct sun-shine conditions.

2.4 Machine Vision

The main problems due to the use of active sensors –besides the pollution of the environment– include inter-vehicle interference amongst the same type of sensors and the wide variation in reflection ratios caused by many different reasons, such as obstacles' shape or material. Moreover, the maximum signal level must comply with some safety rules and must be lower than a safety threshold.

On the other hand, the use of active sensors, which involves the measurement of the alteration of signals emitted by the sensors themselves, possesses some specific peculiarities:

- active sensors can measure quantities in a more direct way than vision. As an example, a Doppler radar can directly measure the relative movements between an object and the viewer, whilst vision can detect movement only through a complex processing of sequences, or –more generally– sets, of images;
- active sensors require less performing computing resources, as they use a considerably lower amount of acquired data.

In many indoor applications, such as the navigation of autonomous robots in unknown settings, vision and active sensors can perform com-

plementary tasks for the recognition of objects, detection of free-space, or in checking for certain specific object characteristics. Unfortunately, when several robots are moving within the same environment, active sensors may interfere with each other, thus decreasing their reliability and usefulness. This problem becomes even greater in outdoor unstructured environments, in which a large number of vehicles could be moving simultaneously, as—for example—in the case of autonomous vehicles travelling on intelligent highways.

Hence, with a massive and widespread use of autonomous sensing agents, the use of passive sensors, such as cameras, obtains important advantages over active ones. These are the cases in which vision becomes of paramount importance. Obviously machine vision can fail—and therefore cannot be used—when humans cannot see (e.g. in foggy conditions or during the night with no specific illumination). In fact, while the use of other sensing agents would have the advantage of extending sensing capabilities besides human possibilities, the use of vision allows the building of a system able to act as the human driver: for example, an active safety system that helps the driver when he fails (e.g. in the lack of concentration).

The recent developments in computational hardware, such as a higher degree of integration and reduction of the power supply voltage, allow the access to machines that can deliver a high level of computational power, with fast networking facilities, at an affordable price. Since the early stages of vision (low-level image processing) are computationally demanding, the availability of low-cost engines helps to solve some basic bottlenecks. Current technology allows the possibility of SIMD-like processing paradigms even in general-purpose processors, such as the new generation of processors that include multimedia extensions (Visual Instruction Set, VIS, in UltraSPARC, MMX in Intel's Pentium, MAX-2 in Hewlett-Packard's PARISCs, MDMX in Silicon Graphics' MIPS, MVI in Digital's Alpha, 3DNow! in AMD's K6-2, and AltiVec in Motorola's PowerPC CPUs).

In addition to this, current cameras include new important features that permit the handling and solving of some basic problems directly at sensor level. For example, image stabilization can be performed during image acquisition, while the extension of camera dynamics allows the removal of the processing required to adapt the acquisition parameters to specific light conditions. The resolution of the sensors has been drastically enhanced, and in order to decrease the acquisition and transfer time, new technological solu-

tions can be found in CMOS sensors, such as the possibility of dealing with pixels independently as in traditional memories. Another key advantage of CMOS-based sensors is that their integration on the processing chip seems to be straightforward.

As a result, not only did this advanced technology promote improved hardware devices, but also triggered off renewed interest in the techniques for the processing of iconic information, generally addressed by the field of Artificial Intelligence, who deals with image interpretation and –more generally– with perception, when the fusion of data coming from other sensors is also integrated.

Nonetheless, when designing a vision system for automotive applications, some important characteristics must be carefully considered.

- ITS systems require faster processing than other applications, since the vehicle speed is proportional to the processing rate. The main problem that has to be faced when real-time imaging is concerned and which is intrinsic to the processing of images, is the large amount of data –and therefore computation– involved. As a result, specific computer architectures and processing techniques must be devised in order to achieve real-time performance. Nevertheless, since the success of ITS apparatus is tightly related to their cost, the computing engines cannot be based on expensive processors. Therefore, either off-the-shelf components or ad-hoc dedicated low-cost solutions must be considered.
- Furthermore, in the automotive field no assumptions can be made on key parameters, for example the illumination or contrast of the scene, which are directly measured by the vision sensor. Hence, the subsequent processings must be robust enough to tolerate these changes and both adapt to different environmental conditions, such as sun (high brightness and contrast due to shadows), rain (extremely high contrast due to reflections), fog (low contrast) and to their dynamic changes, such as transitions from sun to shadows or vice versa (the entrance or exit from a tunnel).

Moreover, other key issues, such as robustness to camera movements and drifts in its calibration must be handled as well.

Nevertheless, although being extremely complex and highly demanding, thanks to the great deal of information that it can deliver, computer vision is

a powerful means for sensing the environment and has been widely employed to deal with a large number of tasks in the automotive field. These tasks include: *Road Following* (which involves *Lane Detection* [85] and *Obstacle Detection*), *Platooning* (the automatic following of a manually driven vehicle by an automatic one [164]), *Vehicle Overtaking* [149], *Automatic Parking*, *Collision Avoidance*, and *Driver Status Monitoring*.

To accomplish these tasks different quantities must be measured and/or patterns recognized before the closing of the control loop. These are the following: the relative position of the vehicle with respect to the lane and the check for obstacles on the path or for known road signs [1] for *Road Following*; the recognition of specific vehicle's characteristics and the computation of the time-to-impact for *Platooning*; the sensing of multiple lanes [156] as well as obstacle detection for *Vehicle Overtaking* and *Collision Avoidance*; the distance amongst parked vehicles for *Automatic Parking*; the position and movements of the driver's eyes and head for *Driver Status Monitoring*.

Machine Vision systems are composed of two modules: the first, used for data acquisition, and the second used to process the acquired data.

Image acquisition

The acquisition of *significant* data is a basic requirement. Many different parameters must be evaluated for the design and choice of an image acquisition device. First of all, the spatial resolution as well as the depth (number of bit/pixel) of the images must be selected; this also includes the selection of color vs monochrome images. Other parameters tightly coupled with the algorithms regard the choice of monocular vs binocular (stereo) vision and the sensors' angle of view. Some systems adopt a multi-camera approach, by using more than one camera with different viewing angles (e.g. fish eye or zoom).

Apart from these, other parameters—intrinsic to the sensor—must be considered. Although the frame rate is generally fixed for CCD-based devices (25 or 30 Hz), the dynamics of the sensor is of basic importance: conventional cameras allow an intensity contrast of 500:1 within the same image frame, whilst most ITS applications require a 10,000:1 dynamic range for each frame and 100,000:1 for a short image sequence. Different approaches have been studied to meet this requirement, ranging from the design of CMOS-based cameras with a logarithmically compressed dynamic [134], to the

interpolation and superimposition regarding values of two subsequent images taken from the same camera [103].

Furthermore, in the automotive field some additional problems have to be faced. The acquisition system is installed on a moving vehicle; therefore image stabilization becomes a key problem. Some expensive devices allow to perform images stabilization directly at sensor level; otherwise, a different solution is to perform it via software [52,104,160], trading the complexity of the sensor with a more powerful computing system.

Moreover, the acquisition system must be robust enough not to interfere with the on-board electronics, especially with telecommunication apparatus.

Finally, for an acquisition system to be of use it must be properly calibrated in order to create the mapping between image pixels and the 3D world.

Image processing systems

Image processing hardware must meet two basic requirements: it has to be powerful enough to support real-time processing, and compact and inexpensive to allow a massive integration on commercial vehicles.

As mentioned, in the early years of ITS applications a great deal of custom solutions were proposed, based on ad-hoc, *special-purpose* hardware. This was motivated by the need for computational power. The most significant solutions were based on SIMD array processors, composed of a large number of extremely simple processors working simultaneously on image windows [23].

After this first stage, when commercial hardware began to deliver sufficiently high computational power at a low cost, *general-purpose* solutions, based on off-the-shelf components, became competitive. In this phase the first MIMD systems [151], composed of a rather small number of powerful, independent processors, were built and installed on prototype vehicles.

Current trends, however, are moving towards a mixed architecture, in which a powerful processor is aided by specific hardware such as boards and chips implementing optical flow computation, pattern-matching, convolution, and morphological filters. Moreover, some SIMD capabilities are now being transferred into the instruction set of the last-generation CPUs, which has been tailored to ex-

exploit the parallelism intrinsic to the processing of visual and audio (multimedia data).

Chapter 3

State of the Art

Different research groups in the academic world as well as within industrial environments are working on the development of intelligent vehicles; in this chapter, after a brief analysis of the most common approaches to Road Following, a short survey is presented about the most advanced projects worldwide that are using vision-based systems.

3.1 Road Following

As discussed in the previous chapter, machine vision is a basic means to sense the environment; a large number of different functionalities for ITS applications have been developed exploiting its characteristics. Among them the most complex and challenging task that received the most attention is *Road Following*. It is based on:

- *Lane Detection*, which includes: localization of the road, determination of the relative position between the vehicle and road, and analysis of the vehicle heading direction;
- *Obstacle Detection*, which is mainly based on localizing possible obstacles on the vehicle's path.

3.1.1 Lane Detection

In most prototype autonomous vehicles developed worldwide, Lane Detection is divided into the two following steps: initially the relative position of the vehicle with respect to the lane is computed and then actuators

are driven to keep the vehicle in the correct position. Conversely, some systems [100, 115] such as ALVINN [34, 80, 116], are not based on the preliminary detection of the road's position, but obtain the commands to be issued to the actuators (steering wheel angles) directly from visual patterns detected in the incoming images. Nevertheless, the knowledge of the lane position can be conveniently exploited by other driving assistance functions.

Although some systems have been designed to work on completely unstructured roads (without painted lane markings) [39, 49, 80] or terrains [101, 133], Lane Detection has been generally reduced to the localization of specific features such as road markings painted on the road surface. This solution can ease the detection, however two basic problems must be faced:

- the presence of shadows (projected by trees, buildings, bridges, or other vehicles), producing artifacts onto the road surface, and thus altering the road texture;
- the presence of other vehicles on the path, partly occluding the visibility of the road and therefore of road markings too.

Since Lane Detection is generally based on the localization of *specific patterns* (lane markings), it can be performed with the analysis of a *single* still image. In addition, some assumptions can aid the detection and/or speed-up the processing:

- due to both physical and continuity constraints, the processing of the whole image can be replaced by analyzing specific regions of interest only (the so-called *focus of attention*), in which it is more probable to find the features of interest [48];
- the assumption of a fixed or smoothly varying lane width allows the enhancement of the search criterion, limiting the search to almost parallel lane markings;
- the reconstruction of road geometry can be simplified by assumptions on its shape; for example some implementations [48, 90, 155] use a clothoid (a curve whose curvature changes without discontinuities) to model the geometry of the road;
- knowledge of the specific camera calibration and the assumption of a precise road model (e.g. a *flat* road without bumps) can be exploited to ease the localization of features and to simplify the mapping between image pixels and their correspondent world coordinates.

Two different approaches have been dealt with in the literature to solve the Lane Detection problem: model-driven and feature-driven.

- A model-based approach consists of a direct match between visual patterns and a model which encodes the characteristics of the road [12, 86], as implemented in VaMoRs [67] or SCARF [40]. In some implementations an initial model is iteratively stretched and re-shaped according to the content of each frame of the acquired sequence; the use of deformable templates (such as LOIS [87] or ARCADE [85]) has been successfully employed for the detection of lane markings, road boundaries and road signs. Also a model-based analysis of road markings has been used to investigate images of intersections in city traffic [54, 89].
Nevertheless, the use of a model-based search approach has several drawbacks, such as: the problem of using and maintaining an appropriate geometrical road model, difficulty in detecting and matching complex road characteristics, and the complexity of the computations involved, since the whole processing is performed on large data structures (images). Moreover, some systems [122] work in the *velocity* domain instead of the *image* domain, therefore using optical-flow techniques in order to minimize the horizontal relative movement of the lane markings with respect to the vehicle. Unfortunately, such a solution requires both the preliminary detection of lane markings and subsequent computation of the optical flow field.
- Conversely, in a feature-based approach the computation intensive processing of images is limited to the extraction of features, and matching the model is left to the following medium- and high-level processing steps, therefore involving the handling of a lower amount of data.

3.1.2 Obstacle Detection

The criteria used for the detection of obstacles depend on the definition of what an *obstacle* is. In some systems determining obstacles is limited to the localization of vehicles, which is then based on a search for specific patterns, possibly supported by other features, such as shape [146], symmetry [164], or the use of a bounding box. In this case, the processing can be based on

the analysis of a single still image, but the approach is not successful when an obstacle does not match the model.

A more general definition of *obstacle*, which obviously leads to more complex algorithmic solutions, identifies as an obstacle any object that can obstruct the vehicle's driving path or, in other words, anything rising out significantly from the road surface. In this case, Obstacle Detection is reduced to identifying *free-space* (the area in which the vehicle can safely move) instead of recognizing of specific patterns.

Due to the general applicability of this definition, the problem is dealt with using more complex techniques; the most common ones are based on the processing of two or more images, such as

- analysis of the *optical flow* field [30, 53], and
- processing of *non-monocular* images [130].

In the first case more than one image is acquired by the same sensor in different time instants, whilst in the second one, different cameras acquire images simultaneously, but from different points of view. Besides their intrinsic higher computational complexity, caused by significant increment in the amount of data to be processed, being based on the processing of multiple images, these techniques must also be robust enough to tolerate noise caused by vehicle movements and drifts in the calibration of the multiple cameras' setup.

The *optical flow*-based [112] technique requires the analysis of a sequence of two or more images: a 2D vector is computed in the image domain, encoding the horizontal and vertical components of the velocity of each pixel. The result can be used to compute ego-motion [66], which in some systems is directly extracted from odometry [91]; obstacles can be detected by analyzing difference between the expected and real velocity fields.

On the other hand, the processing of *non-monocular* image sets requires identifying correspondences between pixels in the different images: two images, in the case of *stereo vision*, and three images, in the case of *trinocular vision* [130]. The advantage of analyzing stereo images instead of a monocular sequence lies in the possibility of directly detecting the presence of obstacles, which, in the case of an optical flow-based approach, indirectly derives from analysis of the velocity field. Moreover, in a limit condition where both vehicle and obstacles have small or null speeds, the optical flow-based approach fails while the other can still detect obstacles. Furthermore, to decrease the intrinsic complexity of stereo vision, some domain specific

constraints are generally adopted.

As mentioned above, a great deal of different techniques have been proposed in the literature and tested on a number of prototype vehicles in order to solve the Road Following problem, but unfortunately in almost all cases, the different issues (e.g. Lane Detection, Obstacle Detection) have been dealt with independently. Obviously an integrated solution leads to both an improved quality of the results and to a faster and more efficient processing.

A very promising approach to these problems is based on the removal of the perspective effect, which is naturally induced by the acquisition conditions. Such a technique has been successfully used for the computation of the optical flow field [94], the detection of obstacles in a structured environment [162], or in the automotive field using standard camera [97, 110, 119] or using linear cameras [131, 154]. Similar techniques [11, 90] (*Image Warping*) have been largely used in the processing of stereo images and used for Obstacle Detection.

3.2 Worldwide Projects

This section presents an overview of the most advanced projects worldwide (in no particular order); their approaches are summarized and compared below.

3.2.1 Research carried out on the MOB-LAB Vehicle (Italy)

The MOB-LAB is an open MOBILE LABoratory suited to develop and test real-time systems, by monitoring the trajectory of a vehicle running on extra urban roads by means of computer vision systems. The MOB-LAB has been developed within the PROMETHEUS project and supported by the Italian *Consiglio Nazionale delle Ricerche* (CNR) through the *Progetto Finalizzato Trasporti 2*.

The MOB-LAB, a FIAT Ducato 18 Maxi, shown in figure 3.1.a, features:

- B/W and Color TV cameras in the front and back of the vehicle for analyzing the external environment;
- a control panel, shown in figure 3.1.b, for alerting the driver;
- two real-time digital image processors with different architectures used for analyzing images at a rate of about 1 image per 200 ms;

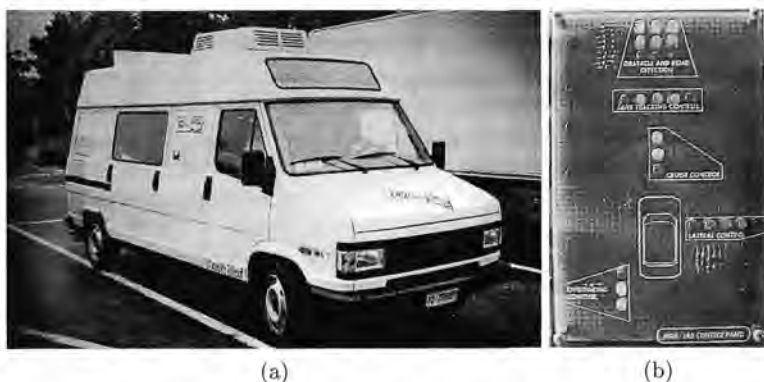


Fig. 3.1 (a) The MOB-LAB vehicle and (b) the led-based control panel used as output to display the results of some experimental functionalities.

- four sensors on board to measure the longitudinal and transverse components of the acceleration;
- one millimeter-wave radar on the vehicle's right side for monitoring the right-hand side of the road;
- two PCs for analyzing data coming from the radar and other sensors.

Several functionalities have been implemented and tested on the MOB-LAB, by different research units within the PROMETHEUS project.

Obstacle and Lane Detection

Road following and detection of obstacles in front of the vehicle are performed by means of a real-time vision system integrating techniques based on visual saliency, with depth cues obtained by stereometry and a-priori knowledge of the road environment. The environment around the vehicle and the dangerous situations are summarized by means of nine different colored leds. This functionality has been achieved by the two Pro-Art research units of Torino: *Istituto Elettrotecnico Nazionale "G. Ferraris"*, *CNR*, and *Centro Studi per la Televisione, CNR* [44].

Lane Tracking Control

Lane tracking is performed through the analysis of data coming from a TV camera and interpreted by means of a special-purpose

massively parallel architecture for real-time image analysis, called PAPRICA (see section 5.1). The result of the analysis is presented to the driver through a set of five different colored leds. More precisely, the leds provide a visual warning to the driver when the car approaches lane borders, becoming increasingly evident when more dangerous positions are taken. This functionality has been achieved by the Pro-Art research unit of the *Università di Parma*, using the low-cost massively parallel board jointly developed by the Pro-Chip research units of the *Politecnico di Torino* and *Università di Parma*.

Cruise Control

This functionality is aimed at estimating the security distance from the vehicle in-front by means of data acquired through a TV camera. The three leds in the control panel tell the driver the current position of the vehicle in-front: no light on means no vehicle; green is for a vehicle at a secure distance; yellow signifies warning; and red means danger. This functionality has been achieved by the Pro-Art research unit of the *Università di Genova* [18].

Lateral Control

The lateral position of the vehicle with respect to a reference line (guard rail or continuous lateral emergency line), is detected through the means of a millimeter-wave radar mounted on the dashboard. Detection of the reference line is made possible by positioning electromagnetic reflectors along it. The data coming from the radar, fused with the vehicle's speed information, are interpreted so as to assist the driver in preventing the vehicle from going off the road, and presented to the driver by means of three different colored leds. This functionality has been achieved by the Pro-Chip research unit *Centro Studi Propagazione ed Antenne*, CNR [41].

Overtaking Control

This functionality is aimed at alerting the driver during an overtaking maneuver if another vehicle is coming from the back. The detection of the incoming vehicle is performed by processing the images acquired from a TV camera looking outside the vehicle from the back glass. The leds in the control panel tell the driver the current position of the overtaking vehicle: green for no vehicles in the back; yellow if a vehicle is present in the back; red if a vehicle has initiated an overtaking maneuver. This functionality has

been accomplished by the Pro-Art research unit of the *Università di Genova* [149].

Car attitude Control

Different low-cost sensors inside the vehicle measure the longitudinal and transversal components of the acceleration. Data coming from the sensors are interpreted and presented to the driver by means of visual information. The sensors have been designed and manufactured by the Pro-Chip research units of the *Università di Brescia* and *Università di Modena*.

The MOB-LAB was officially presented and demonstrated during the final PROMETHEUS Board Member Meeting; its availability allowed the development, testing, and tuning of a number of different approaches used for autonomous navigation.

3.2.1.1 Lane and Obstacle Detection on the MOB-LAB vehicle

This paragraph presents the Obstacle and Lane Detection systems developed at the *Istituto Elettrotecnico Nazionale "G. Ferraris"*. The Obstacle and Lane Detection functionalities have been developed to work in a structured environment. The Lane Detection functionality relies on the painted lane marking's presence and Obstacle Detection depends on the Lane Detection module's result.

The whole process is based on the use of images that come from a single grey-level CCD camera. In a structured highway or highway-like road scenario, two different models are used:

- the road is always located in the center low portion of the image;
- due to the perspective effect, the road almost takes a triangular shape.

The image is horizontally parsed starting from the bottom and looking for high brightness variations that are likely to detect a marking. The detected markings are followed towards the upper part of the image and a check is carried out with the triangular model to discard bad results. Also the temporal correlation is used to validate the result and reduce the area of the image to be processed.

Obstacle Detection is performed by limiting the processing to the portion of image that is assumed to represent the road. This area of the image

is processed and borders that could represent a potential obstacle are looked for. Three different states for obstacles are considered:

- (1) alert: all potential obstacles are detected;
- (2) tracking: potential obstacles are matched with the result related to the previous frame and tracked;
- (3) identification: each hypothesis regarding obstacles is validated using temporal correlation and a specific obstacle template.

The system has been implemented on the MOB-LAB vehicle using an Imaging Technology computing engine; currently no direct actuation on the vehicle is performed, but only warnings are issued to the driver.

3.2.2 *Research carried out at the Centro Ricerche FIAT (Italy)*

3.2.2.1 *The LAKE autonomous vehicle*

The LAKE autonomous vehicle is a Lancia Thema passengers' car developed by the *Centro Ricerche FIAT* and is aimed at developing and testing both guidance assistance functionalities and automatic steering capabilities [4].

LAKE has been developed to be used on highways or sufficiently structured roads. The basic idea is to prevent unintentional or harmful maneuvers tracing the position of the vehicle with respect to the lane and, when needed, taking control of the steering wheel (see figure 3.2) or giving the driver an acoustic or a mechanic alert. An additional input device returns the position of the indicators lever, therefore allowing the system to foresee the driver's intentions in lane changing.

The main input system is a single standard camera installed behind the upper part of the windscreen. The camera is equipped with a wide-angle lens and works at the rate of 25 Hz. Incoming images are digitalized at 512×512 pixel resolution. Only the part of the images correspondent to the nearest 12 meters of road in front of the vehicle is processed; experimental tests demonstrated that this allows to safely maneuver the vehicle in highways and even in belt ways or ramps with a bending radius higher than 50 meters.

The whole processing relies on the presence of road markings and is based on the following assumptions:

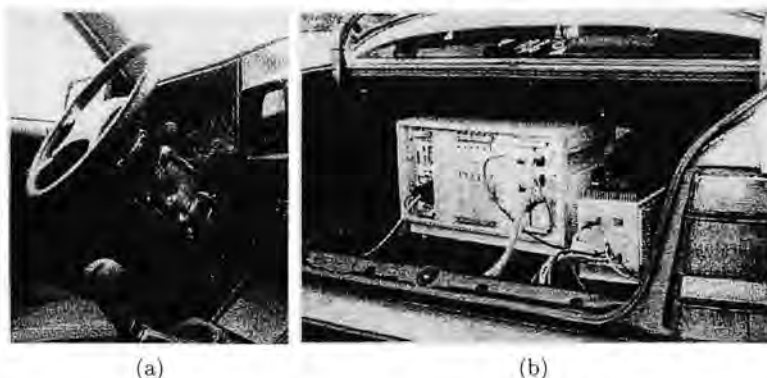


Fig. 3.2 The LAKE vehicle: (a) automatic steering device and (b) computing engine (published courtesy of Giulio Vivo – Centro Ricerche FIAT).

- the portion of lane markings next to the vehicle can be approximated by straight lines: a linear model for the road can be used;
- the road in front of the vehicle is flat or, rather, its slope is negligible: it is possible to fully determine the relationship between the image domain and the real world.

Lane markings are seen as bright lines on a darker background having the same vanishing point. Hence, the first part of the processing lies in extracting the borders in the image using a Sobel operator [139]. Then the directions of these borders are determined and a histogram of directions is computed. Peaks in this histogram correspond to straight lines in the image and, since the Sobel operator applied to lane markings produces two edges, a lane marking causes the presence of two near peaks in the direction histogram, therefore permitting the rejection of false lines induced by noise or by the presence of brightness discontinuities in the asphalt.

Once a lane marking is identified, it is modeled using a linear model and, thanks to a phase filtering, possible positions in the next image are also computed. This allows processing of only the part of the incoming frame where it is most likely to find the detected lane marking.

The system is composed of several VME boards (summarized in figure 3.3) supervised by a 486DX2-66 based Personal Computer and can be thought of as a pipeline structure:

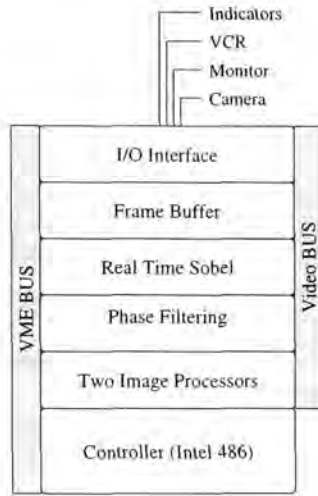


Fig. 3.3 The LAKE computing engine.

- the acquisition task is performed by an Image Technology 150/10 I/O adapter;
- two different adapters act as frame grabbers;
- the extraction of borders in the image is performed by a Sobel operator;
- a border following algorithm is executed thanks to the use of a phase filtering adapter;
- two image processing adapters are used for the distinct detection of right and left lane markings;
- the acoustic and visual output are managed by the PC.

Since the LAKE vehicle has been developed mainly for testing and as a demonstrative vehicle, using a switch it is possible to choose the system behavior and to test different solutions and man-machine interactions. According to this selector the LAKE vehicle features three different kinds of output.

Visual output: the LAKE vehicle is equipped with an on-board monitor; detected lane markings are superimposed on the acquired image in

a brilliant color (see figure 3.4), also the distances are shown (in cm) between the lane markings and the vehicle's wheels.

Acoustic output: when the vehicle is too near to the left or to the right lane marking an acoustic warning is issued to the driver unless the position of the indicator lever shows the the driver's intention to change lane.

Direct actuation on the steering wheel: the result of the computation is fed to a specific actuator (see figure 3.2.a) that controls the steering wheel; in this case the following three different kinds of actuations are possible.

Warning: the vehicle is controlled by the human driver, the actuator simply contrasts but does not prevent wrong maneuvers.

Correction: the vehicle is still controlled by the human driver, but, when needed, the system can override the driver keeping control of the steering wheel and adjusting the trajectory.

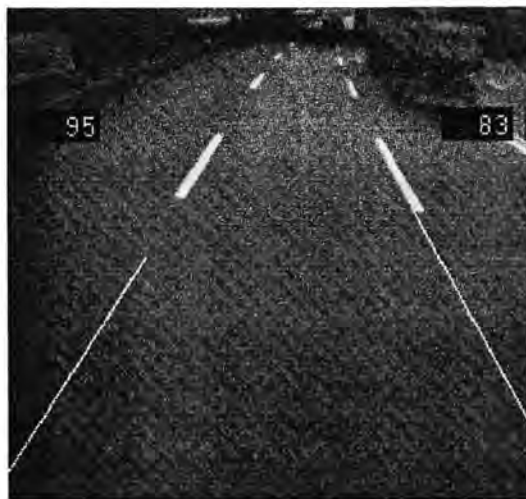


Fig. 3.4 Visual output on the on-board monitor of the LAKE vehicle: the bright lines in the lower part of the image indicate the computed lane, the two numbers in the upper part of the image show the distance of the left and the right markings from the vehicle (published courtesy of Giulio Vivo – *Centro Ricerche FIAT*).

Full Control: the system keeps full control of the steering wheel tracing and maintaining the position of the vehicle with respect to the lane.

3.2.2.2 The SAVE vehicle

The SAVE (System for effective Assessment of the driver's state and Vehicle control in Emergency situations) vehicle (shown in figure 3.5) is aimed at helping the driver in emergency situations [35]. SAVE is capable of both detecting driver impairment and performing an emergency maneuver. This last task is accomplished by maintaining a safe distance with respect to other vehicles or potential obstacles, performing lane changes, and decelerating.

Safe distance: the SAVE vehicle is able to detect, locate, and track any obstacle in front of the vehicle using a laser radar located inside the front bumper.



Fig. 3.5 The SAVE vehicle (published courtesy of Giulio Vivo – *Centro Ricerche FIAT*).

Lane change: a vision system similar to the one described for the LAKE vehicle has been implemented into the SAVE vehicle. Since the lane change maneuver has to be as *normal* as possible and must not to excessively disturb the traffic, the system computes a steering angle correspondent to a *soft* lateral speed (namely ≤ 0.5 m/s) and feeds it to the steering wheel actuator.

Deceleration: the SAVE vehicle also features actuators for the throttle and the brake therefore allowing control of the vehicle's longitudinal speed.

Starting from a speed of 125 km/h, the SAVE vehicle is able to perform a full stop emergency maneuver in less than 30 s in a three lane road, with a lane-change angle of 0.65 degrees and an average lateral speed of 0.29 m/s.

3.2.3 Research carried out at the Universität der Bundeswehr (Germany)

A large number of vehicle prototypes have been developed at the *Universität der Bundeswehr*.

In 1985, their first autonomous vehicle VaMoRs (a 5 ton van [67]) was tested in a real scenario, at a speed of up to 100 km/h, in an empty highway. It used machine vision to maintain both longitudinal and lateral control of the vehicle. In 1988, during the first board member meeting of the PROMETHEUS project in Torino, the autonomous vehicle VITA (a 7.5 ton van [46]) was demonstrated: it had *stop and go* capabilities and was capable of convoy driving behind another vehicle (platooning). Both these vehicles were equipped with the *UBM* vision system [67]. This is a two-axis pointing system with high bandwidth and inertial stabilization, including other sensors like: three orthogonally-mounted accelerometers, a wheel position encoder able to act both as an odometer and speedometer, and, in the case of VaMoRs, a GPS receiver for a coarse initial guess of the vehicle's position.

In 1994 the next generation vehicles VITA II, developed by *Daimler-Benz* in cooperation with the *Universität der Bundeswehr*, and VaMP were the only ones capable of driving autonomously during the final demonstration of the PROMETHEUS project. The most noticeable evolution in these systems was a move towards a massive use of Transputers (the VaMoRs-P, twin of VITA II, was a Mercedes 500 SEL equipped with 60 Transputers,

also capable of autonomous overtaking [98]); this decision allowed the reduction of the processing system's size to about one third, and –at the same time– increased noticeably the processing power.

These systems were based on a *bifocal* vision system, in which several cameras with different focal lengths were used for separate targets. Road Detection was achieved thanks to the use of a wide-angle camera that allowed large viewing, while Obstacle Detection was performed using a small-angle camera with a good resolution further away.

Thanks to the technological evolution, the Transputer processing system has now been partly replaced by a cluster of three PCs (dual Pentium II) connected via a Fast Ethernet-based network. This cluster is capable of carrying out ten times the amount of computing performance on specific tasks [93].

The functionalities of the latest prototypes are summarized in the following.

Road Detection: it is based on a viewing range of up to 100 m and on a specific model for the road; several lanes may be tracked in parallel. The model used for the lane markings is a *clothoid*, that allows the definition of markings using the concept of *curvature*. In a clothoid the curvature depends linearly on the curvilinear reference l :

$$C = C_0 + \frac{\partial C}{\partial l} \times l = C_0 + C_l \times l$$

Therefore C_l is constant, and the knowledge of C_l and C_0 allows the full localization of lane markings and computation of the other following parameters: lane width, number of lanes in the look-ahead range, lateral offset within the lane, lateral speed with respect to the lane, and steering angle.

The use of a Kalman filter [81] permits reaching the working rate of 25 Hz and speeds of up to 100 km/h.

Obstacle Detection: it is based on an edge detection process and obstacle modelization; the system is able to detect and track up to twelve objects around the vehicle. The following obstacle variables are continuously computed: distance, direction, relative speed, relative acceleration, lateral position, lateral speed, and size.

Other Functionalities: other basic functionalities were developed and evaluated, such as: *cross-roads* in which both the width and angle

of intersecting roads are estimated [47], *overtaking* initially demonstrated in Paris [152], *platooning* directly deriving from the Obstacle Detection functionality [25].

The most innovative idea in the input system is the use of a dynamic multi-focal vision approach (EMS, Expectation-based Multi-focal Saccadic). Initially multi-focal capabilities were obtained thanks to the use of different cameras, whilst now, a new multi-focal compound camera has also been developed. It is composed of a number of objectives and sensors [125]:

- two wide-angle lens sensors covering around 130° in the horizontal plane, therefore allowing both a front and partly lateral view of the environment;
- a high resolution color camera featuring a mid-wide angle lens covering the central zone of the viewing area;
- an optional high sensitivity imaging sensor featuring a very small-angle lens providing additional capability of high resolution in small fields of view.

The viewing direction of the compound camera is under the control of the processing engine, and is modified according to the result of the computation in order to track specific image features.

In order to cope with the problems associated to managing multiple video streams, each optical sensor feeds a different node in the PC's cluster.

The whole system is based on the fourth dimension concept [48]: the basic idea is that, since the world has four dimensions (assuming time as the fourth), the models used for describing the world must also be four-dimensional. Since the world is not static (assuming that the vehicle is moving), the internal world model must also change according to both the computation and acquired data. In fact, in correspondence to each acquisition process, a match between the acquired data and the predicted situation is carried out, allowing to closely track the changes in the external world.

The processing system is able to keep control of the steering wheel, throttle, and braking system, therefore enabling full autonomous driving. The maximum speed is computed according to the horizontal curvature of the lane. The system is also capable of autonomously deciding and performing an overtaking maneuver if the leading vehicle is too slow. The passengers can act on specific parameters for changing the system's behav-

ior, such as to a more sportive driving style, a platooning state, or allowing the vehicle to turn into a cross-road.

3.2.4 Research carried out at Daimler-Benz

Daimler-Benz has a long experience in autonomous driving and achieved a variety of experimental cars. As mentioned in section 3.2.3, their first demonstrator VITA was followed by VITA II, (a Mercedes-Benz S-class) that demonstrated fully autonomous driving on public highways including lane changes [152]. It was equipped with 18 cameras looking forwards, behind and sideways in order to allow a 360 degrees view. Inverse perspective mapping (see section 4.3.1) was used to survey the neighboring lanes and detect obstacles in front of the vehicle. Built in cooperation with the *Universität der Bundeswehr* and the *Ruhr-Universität Bochum*, VITA II was able to recognize other traffic participants, the road's course as well as the relevant traffic signs. In addition, it was provided with a behavioral module that was able to plan and perform overtaking maneuvers in order to maintain the desired speed; in this case, the side- and rear-looking cameras were used to ensure safety of these maneuvers.

Simultaneously to the PROMETHEUS demonstrators, the Mercedes-Benz T-model OSCAR was built to investigate vision algorithms and control schemes for robust and comfortable lateral guidance on highways at high speeds. The algorithms were based on the standard lane tracking approach developed by the research group of the *Universität der Bundeswehr* in a joint project. Based on the Transputer technology of the early nineties, OSCAR drove about 10,000 km on public highways with maximum speeds up to 180 km/h using conventional as well as neural controllers [108]. OSCAR tracked not only the markings, but also looked for structures parallel to the lane. From the algorithms used in this car, a lane departure warning system has been created that has recently been announced as a commercially available lane-departure warning system [163].

In 1995, *Daimler-Benz* completed the work on the OTTO-truck [63]. With the AHS idea in mind, this truck was designed to follow a specific leader with minimum distance. To accomplish this task, OTTO measures the distance to the vehicle in front by looking at known markers. An infrared light-pattern is used as well as two checker-board markings. In order to reach minimum distance and handle leader's emergency braking, its acceleration is transferred to the follower by means of a communication

link. Recently, OTTO has been replaced by a heavy duty truck (40 ton) within the European Chauffeur project.

The UTA project (Urban Traffic Assistant) aims at an intelligent stop and go for inner-city traffic. The use of stereo vision by means of a pair of cameras with a base width of 30 cm permits to obtain 3D information in real time. In addition, the UTA demonstrator is able to recognize traffic signs, traffic lights and walking pedestrians as well as the lane, zebra crossings, and stop lines. The driver is acoustically informed about the presence of significant objects [64].

3.2.5 *Research carried out at the Fraunhofer-Institut für Informations und Datenverarbeitung (Germany)*

The *Fraunhofer-Institut für Informations und Datenverarbeitung* research group possesses the main target of developing a driver assistance system capable of facing complex scenarios like urban traffic environments.

3.2.5.1 *Lane Detection*

One of the main achievements is the development of a vision-based system named ROMA (ROad Markings Analysis [54]) that features the following functionalities:

Road borders detection: a contour-based method, that does not rely on road markings, is used. A dynamic road model permits the processing of only small portions of the acquired (grey-level) images therefore enabling real-time performance. Actually, only straight or small curved roads without intersections are included in this model. Images are processed using a gradient-based filter and a programmable threshold. The road model is used to follow contours formed by pixels that feature a significant gradient direction value. A dedicated hardware, the MiniVISTA-System [65], is used to perform the low-level image processing.

Intersection detection: since road intersections can be distinguished by using either dashed or continuous markings or even no markings at all, different strategies have been used.

The first approach relies on the presence of a continuous border marking. A search for markings that cross-over is carried out. When such a marking corresponds to a sufficiently large gap in

the border marking, a potential intersection is considered. A digital road map [105] as well as a Kalman filtering are used to confirm this data-driven hypothesis and reduce the measurement noise.

The second approach also permits the handling of intersections marked by dashed lines. The images are processed using a gradient-based filter in order to find straight lines segments. These segments are considered so as to determine whether they form a *dash* of a road marking; during this step a dash list is built. Considering smoothly varying lane markings and starting from the assumption that dashes belonging to the same lane marking should have a similar width, dashes are grouped together to reconstruct the lane markings.

Some a-priori knowledge about the shape of marking lines is used to detect intersections. In fact, in Germany, the markings used to distinguish an intersection are narrower than the ones used to mark the road.

This approach has been implemented using both the MiniVISTA-System and a Transputer-based architecture.

Another approach to Lane Detection was developed more recently [126]. In this case, lane markings are supposed to be parallel and are modeled using a clothoid approximated by a third order polynomial:

$$y_l = \pm \frac{b}{2} + \frac{1}{2} \times C_{h_0} \times l^2 + \frac{1}{6} \times C_{h_1} \times l^3$$

where:

y_l represents the y coordinate of a lane marking point in the world system (assuming a y -axis *perpendicular* to lane markings);

b is the road width (the $\pm$ includes both left and right markings);

C_{h_0} is the local curvature;

C_{h_1} is the curvature variation;

l is the curvilinear coordinate.

The position of lane markings in an image n of a sequence is guessed using the following relationships:

$$\begin{aligned} y_l(t_n) &= y_l(t_{n-1}) + \nu \times \varphi \times \Delta t \\ C_{h_0}(t_n) &= C_{h_0}(t_{n-1}) + C_{h_1} \times \Delta t \end{aligned} \quad (3.1)$$

where:

- ν represents the known speed of the vehicle,
- φ is the vehicle's yaw with respect to the markings, and
- t represents the time.

This allows the reduction of the portion of the image to be processed.

Images are scanned along horizontal lines whose density is proportional to the distance from the vision system. A complex gradient filtering is used to determine edge pixels. Only the border points closest to the state predicted using equations (3.1) are considered.

Other parameters, such as the overall distance from the predicted state or tolerance factors for road shape, are used to validate or reject the result.

3.2.5.2 Obstacle Detection

The Obstacle Detection functionality is based on the use of an optical flow technique in conjunction with data coming from an odometer [91, 158].

Thanks to the MiniVISTA hardware and two PowerPCs, the optical flow vectors are computed from spatio-temporal derivatives of grey-level images. In order to reduce the size of the computed data, estimated vectors are grouped together using similarity and closeness as criteria. Another grouping step allows the determination of vectors that belong to the same obstacle. For this purpose, a plane parallel to the road surface is considered and all the vectors that are classified as belonging to an obstacle are projected onto this plane. Using assumptions concerning the obstacle width the projections are matched each other in order to determine the ones that belong to the same obstacle.

3.2.6 Research carried out at the Laboratoire Central des Ponts-et-Chaussées de Strasbourg (France)

This section describes the work developed in France by the research group of Strasbourg [32, 33, 49, 50].

The main aim of this work is not the development of an autonomous vehicle but the design of an image processing method to be implemented on an automatic repainting module for road markings recovery. This target not only involves lane markings detection but also requires the extraction of specific features, such as their width or modulation, as well as the identi-

fication of other planar road signs that cannot be repainted automatically.

The currently used repainting vehicles require human operators and work at a low speed (10 km/h), this also applies to complex traffic scenarios. Even if this processing is partially automated, the monitoring and parameters (such as the dashes modulation or the line width) still have to be under the control of a human operator. In order to improve working safety and processing speed, the development of an automatic repainting module (MRA from the French acronyms *Module de Repassage Automatique*) is one of the aims of this research group. Basically, it requires the following tasks:

- road identification;
- markings recognition;
- polygonal representation of markings;
- actuation on the repainting system.

A 3-CCD color camera is installed on the MRA; it provides images of the working environment up to 50 m ahead of the vehicle. Only speeds up to 20 km/h have been considered, therefore allowing both the processing of a reduced portion of the acquired image (namely up to 30 m in front of the vehicle) and use of a slow processing rate (1 image/s).

The first step of the processing identifies the part of the image that corresponds to the road, therefore speeding up the following feature extraction work. Since in this application also non-structured roads are considered, the road identification cannot be based on markings recognition. At the same time, problems (such as shadows, defects, and dirt) must also be faced. The basic assumption used is that there should always be a chromatic contrast between road and off-road, at least in one color component. To separate the two components, the concept of *chromatic saturation* [77] is used (figure 3.6 shows a road view and its saturated version). It represents a measurement of how much a single color component in a pixel prevails upon the other two and it does not strictly depend on the brightness of the pixel (namely the chromatic saturation is a measurement of how colored or grey is a pixel). A neural network approach has been originally chosen for this step, while a novel and simpler approach that does not rely on the use of neural networks is now under evaluation [33].

The road area is scanned using an adaptive threshold to identify painted markings and signs. Thanks to a Radon-transform [76,92] based approach, the detected signs are matched and stored using a polygonal representa-

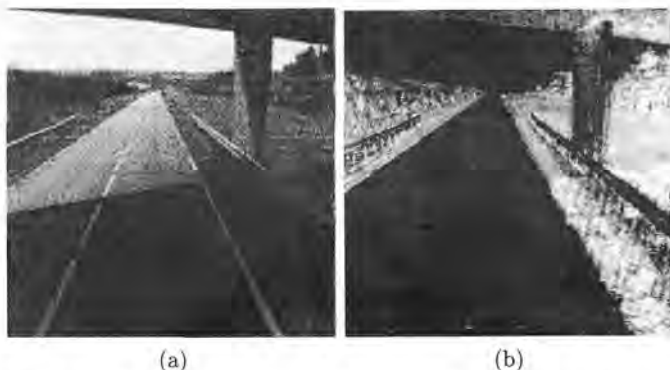


Fig. 3.6 A road image (a) and its saturation (b) (published courtesy of Pierre Charbonnier – *Laboratoire Régional des Ponts et Chaussées de Strasbourg*).

tion (one or more trapezoids). In order to improve the robustness of the approach, this last phase is performed using the luminance component of the acquired image.

Finally the computed trapezoids are grouped to determine the lane markings and estimate their size. This step is performed using the Hough transform [73].

3.2.7 Research carried out at the Defence Evaluation & Research Agency (Great Britain)

The research areas investigated at the *Defence Evaluation & Research Agency* include feature detection, image filtering, motion segmentation, 3D reconstruction, autonomous vehicle research and AI research aimed at the application to land vehicles [43]. In particular, the following prototype has been developed.

The ROVA Experimental Vehicle: the ROVA testbed vehicle is an autonomous campervan equipped with:

- control actuators for steering, accelerator and brakes;
- several state sensors;
- interfaces between the actuators/sensors and Transputer links;
- a steerable camera head with two video cameras;

- Transputer-based image processing systems:
 - Transtech i860/Transputer-based system;
 - Parsytec PowerPC/Transputer-based system;
- VME-based dedicated hardware - the Roke Manor hardware feature detector;
- a Sun host computer.

The following systems were implemented on the ROVA vehicle.

ASSET-2 (A Scene Segmenter Establishing Tracking v2):

is a complete real-time vision system for segmentation and tracking of independently moving objects. The main feature of ASSET-2 is that it does not require any camera calibration. It tracks down objects and is capable of handling correctly occlusions amongst obstacles, and automatically tracks down each new object that enters the scene.

ASSET-2 initially builds a sparse image flow field and then segmentates it into clusters that feature homogeneous flow variation. Temporal correlation is used to filter the result, therefore improving accuracy [138].

ALTRUISM (Automatic Local Three-dimensional Representation Using Image-based Scene Measurements):

the main aim of ALTRUISM is to three-dimensionally reconstruct the scene, determining the free-space and position of obstacles.

Another image processing module named DROID is in charge of providing the three dimensional scene information for this task. It provides a list of measurements regarding the three dimensional scene made at specific positions in the image. DROID also produces an estimate of the camera's own motion in the world, therefore enabling the solution of the scale/speed ambiguity. It is a structure-from-motion system and works by tracking two-dimensional features over time, using a Kalman filter for the three-dimensional tracking of each point [81].

ALTRUISM starts by finding a recursive least squares fit of a given model to the 3D tracked features, giving an estimate of the three-dimensional parameters of the road. Using this estimation, DROID is able to determine the three-dimensional

features that lie on the driveable surface and compute a *radial map* of this surface. This map is temporally filtered to reduce noise and to interpolate missing data.

DROID-ALTRUISM has been implemented as a real-time system on the ROVA PowerPC/Transputer image processing equipment but the result of the processing is still not fed to the steering wheel actuators [137].

Integrated Real-Time Motion Segmentation and 3D Interpretation: it integrates the real-time motion segmentation system (ASSET-2) and the 3D reconstruction/interpretation system (ALTRUISM).

The ASSET-2 system allows the recognition of moving objects from the background. Once all the moving objects have been determined, ASSET-2 produces a list of features that belong to the static part of the scene.

This enables the DROID system to run successfully on the background part of the image sequence. Finally its output is interpreted by ALTRUISM, in order to give warning of potential obstacles.

ALERT (Automatic Lane Extraction in Real-Time): this system includes lane tracking and longitudinal automatic control integrating computer vision and radar.

ALERT processes sequences of images and tracks the lane markings, using a combination of local inter-marking interpolation and global extraction running on two PowerPCs.

Horizontal line segments are tracked within the lane and the ones associated with the bottom of the next vehicle are determined. ASSET-2 is used to detect overtaking vehicles.

3.2.8 Research carried out at Carnegie Mellon University (United States)

The research team of *Carnegie Mellon University* is mainly focused on the two following activities:

- experimenting with trinocular vision and range finders for off-road situations, involving Army HMMWVs;
- enhancing both safety and throughput in highway traffic.

Since 1984 [157] these groups have developed a number of different systems implemented on several experimental vehicles, called the NavLab series [78, 79, 147, 148].

SCARF (Supervised Classification Applied to Road Following):

this system is based on the use of a color camera and on the assumption that the road would be homogeneously colored. It relies on the use of a road model and has the capability to also handle unstructured roads [39, 40].

YARF (Yet Another Road Follower): the use of a number of models for each automotive environment allows this system to face different situations such as highways, freeways, and also rural roads. When working on structured roads the use of color information allows for the classification of lane markings [89].

ALVINN (Autonomous Land Vehicle In a Neural Net): this system is based on a neural net approach. Thanks to a training phase based on a large set of images, the system is able to follow the road. The input images are filtered to enhance the road and are reduced (30×32) to speed up the processing [80].

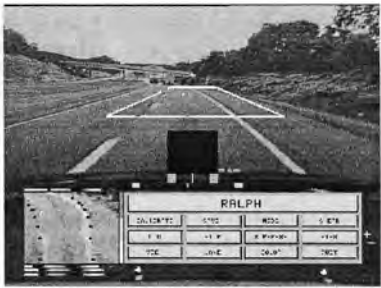
Nevertheless, the latest and most promising approach is the RALPH system (Rapidly Adapting Lateral Position Handler [119]). The RALPH system has been implemented on the NavLab 5 vehicle (see figure 3.7.a). It is equipped with: a portable computer for processing, a color CCD camera installed behind the windscreen near the rear-view mirror, a range finder used for Obstacle Detection, and a GPS system for determining the vehicle's speed.

The Obstacle Detection functionality is based on the use of the range finder, whilst Lane Detection is performed by means of artificial vision. Since great part of the acquired image is not relevant to driving tasks, only a small portion of the image is actually used: more precisely, the part corresponding to the road portion about 20 to 70 meters ahead of the vehicle, depending on the vehicle's speed and obstacles presence. An IPM like transform (see section 4.3.1) is used to homogenize this portion of the image, which is also subsampled and reduced to 30×32 pixels (see figure 3.7.b).

The determination of the curvature is carried out according a number of possible curvature models (for a specific road template), which are used for *straightening* the subsampled image. The result of this straightening



(a)



(b)

Fig. 3.7 (a) The NavLab 5 a Pontiac Transport and (b) output of the RALPH system: the bright trapezoid detects the portion of the image used for processing; the small image (30 × 32 pixels) in the bottom left hand corner of the picture represents a partial result of the computation (published courtesy of Dean Pomerleau – Carnegie Mellon University).

determines how the model fits the actual road configuration. In fact, in a fully straightened image the lane markings are vertical lines and a simple histogram, which is built adding the brightness value of the pixels along the vertical axis, presents sharp discontinuities.

To adapt the system's behavior to different road scenarios, various road templates are used. Four methods have been implemented to adapt the template to the current road environment:

- a switch is used to induce RALPH to determine the template starting from the current image;
- a library of templates is used: the template possessing the highest correlation with the current image is selected;
- the current template is gradually evolved to follow small road changes;
- abrupt scene changes are handled also using road appearance far ahead of the vehicle (70 to 100 meters).

Once the most appropriate model has been found, three parameters are computed and used to control:

- the vehicle lateral offset,
- the vehicle yaw relative to the road's centerline, and
- the upcoming road curvature.

Since the whole vision process does not require the extraction of features from the image and thanks to the reduced size of the processed image, a simple PC-486 with an acquisition device is used.

The processing system is able to keep control of the steering wheel and the throttle. No actuation is performed on the braking system: when an obstacle that is too close is detected, the throttle is simply released.

Using the results of the processing, the position of the lane center at a fixed time (around 1.6 s) ahead of the vehicle is computed. According to this result the steering angle of the vehicle is handled to head the vehicle to that point. Other enhancements, such as a Kalman filtering, are used to improve the estimates and smooth vehicle movements.

In 1995, the RALPH system was tested on a journey (*No Hands Across America*) from Pittsburgh (Pennsylvania) to San Diego (California). On this trip, the NavLab 5 was able to drive itself 98% of the way at an average speed of 102 km/h.

3.2.9 Research carried out at The Ohio State University (United States)

The Ohio State University Center for Intelligent Transportation Research has developed three automated vehicles (see figure 3.8) demonstrating:

- advanced cruise control,
- automated steering control for lane keeping,

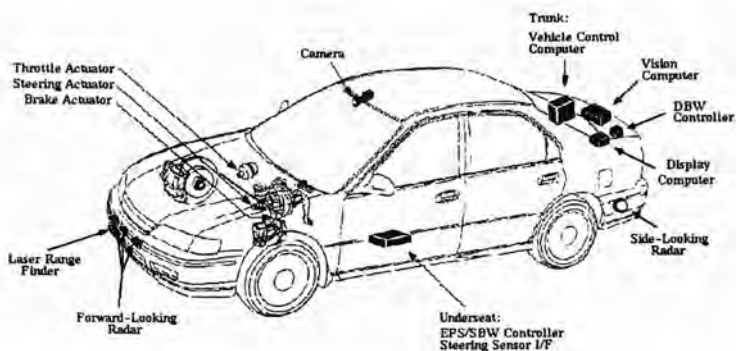


Fig. 3.8 The equipment and its layout of the autonomous vehicles developed at *The Ohio State University Center for Intelligent Transportation Research* (published courtesy of Ümit Özgüner and Keith Redmill - *The Ohio State University Center for Intelligent Transportation Systems*).

- automated stopping,
- lane changes,
- and other autonomous behaviors in presence of other vehicles.

These vehicles are equipped with a number of different sensors in order to implement both data fusion and fault detection techniques; more specifically:

- a vision-based system,
- a radar system (to determine the lateral position within the lane, exploiting reflective lane stripes),
- a scanning laser range finder (for Obstacle Detection),
- other sensors like side-looking radars and an angular rate gyroscope.

Two of these vehicles were demonstrated at the National Automated Highway System Consortium (NAHSC) 1997 Technical Feasibility Demonstration in San Diego, California, during August 7-10, 1997 [55, 113].

3.2.9.1 Lane Detection

In this paragraph the vision-based portion of Lane Detection is described. The Lane Detection system processes grey-scale images acquired using a

single CCD camera positioned in place of the central rear view mirror as high as possible. The algorithm assumes a flat road with either continuous or dashed bright lane markings. The history of previously located lane markings is used to determine the region of interest, thus reducing the portion of image to process. The algorithm extracts the significant bright regions from the image plane and store them in a vector list. Qualitative parameters such as the convergence of the lines at infinity or the lane width, known or estimated, are used to extract from the list the candidate lane markings. Finally, in order to handle also dashed lines, a low-order polynomial curve is fitted to the computed vectors.

When available both, right and left markings are used to estimate the center of the road; otherwise an estimation of the lane width and the only visible marking are used.

The algorithm has been implemented on a Visionex Smarteye I, which is a dedicated image processing system based on the Texas Instruments TMS320C30 digital signal processor and equipped with two monochrome framegrabbers able to buffer 512×480 8-bit images. The processor runs at 40 MHz. To speed up the computation both framegrabbers are used: while the CPU processes an image, the next frame is being captured by the other framegrabber. The overall working rate is approximately 17 Hz.

The vision system's accuracy is about ± 5 cm at a lookahead distance of approximately 6 meters; results are both fed to the control system using a RS232 and displayed on the image output of the Smarteye.

During the NAHSC '97 Demonstration, the system was tested for over more than 1500 miles both on the interstate I-15 and on tracks with tight curves: it demonstrated to be robust and reliable in a number of different conditions (new or worn asphalt, concrete, wet road...) [124].

3.2.10 *Research carried out at the University of Michigan (United States)*

3.2.10.1 *The ARCADE system*

The ARCADE system (Automated Road Curvature And Direction Estimation) has been developed for road boundary determination.

A first approach was to approximate road boundaries using straight lines with the same vanishing point. The edges of the road boundaries were computed, compared to a given model, and used to determine a new

vanishing point to be used for the processing of the next image. Obviously the system needed an initialization phase to determine the first vanishing point.

A second and more robust approach [88] is based on the use of two specific assumptions: the road and its lanes can be approximated by circular arcs* and the road in front of the vehicle is flat. The edges of the image are computed looking for brightness transitions in a selected row of the image. Since the result could be affected by noise, the result is filtered using a least median-square estimation technique that allows facing situations in which up to 50% of the edges are due to noise.

3.2.10.2 The MOSFET vehicle

MOSFET (Michigan Offroad Sensor Fusing Experimental Testbed) is an autonomous vehicle developed at the *Electrical and Computer Engineering Department of the University of Michigan* for navigating in unknown terrains [102]. MOSFET is equipped with three color cameras and a bank of sonars. The system extracts information from each individual sensor and then fuses them, also using the temporal correlation amongst sensed data.

The vision system relies on the presence of bright lane markings. A color segmentation algorithm is used to transform all the acquired color images into grey-scale ones. The algorithm is based on a-priori knowledge of the typical markings' parameters and of the background. Furthermore, it is able to maximize the contrast between the lane pixels and background.

The two side-looking cameras are used to determine the position and orientation of MOSFET with respect to the left and right lanes. An intensity profile of three rows from each grey-scale image is matched against a model of an ideal marking; the model is also adapted to reflect the new situation.

Conversely, the forward-looking camera is used to determine the road shape in front of the vehicle. In this case, a fitting procedure of a lane marking model allows the determination of the lane position and orientation. Lane markings are modeled as parabolas $y = ax^2 + bx + c$, where a represents the curvature, b the orientation, and c the offset. Since the markings are assumed to be parallel, they share the same values for a and b . A simplified Hough transform is used to accomplish the fitting procedure.

*More precisely $x = \frac{1}{2} \times k \times y^2 + m \times y + b$.

3.2.11 *Research carried out at the Massachusetts Institute of Technology (United States)*

At the *Massachusetts Institute of Technology* researchers of the Intelligent Transportation Research Center are working on various issues related to ITS. A feature of this research group is the integration of different sub-systems from integrated-circuit chips to network architectures used for heterogeneous traffic information; some projects are described in the following.

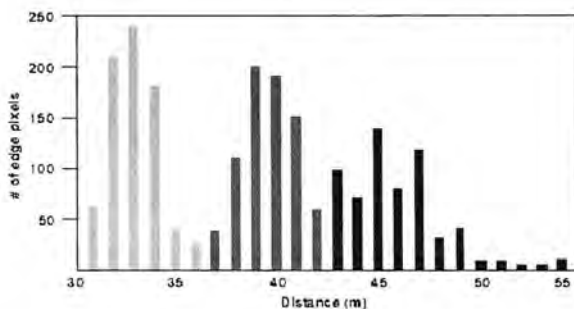
Adaptive Dynamic Range TV Camera: conventional TV cameras do not have enough intensity dynamic range for ITS and other applications which require high reliability. The adaptive dynamic range TV camera chip can increase its dynamic range by a factor of 100. The range can also be dynamically adjusted from image-frame to image-frame to adapt to the environment (i.e. to obtain a better intensity-resolution when a wide dynamic range is not required). The chip is made by using the CMOS technology instead of the conventional CCD technology in order to make system-on-a-chip architecture possible. On-chip parallel analog-to-digital converters are also implemented.

Array processor chips: since real-time early vision requires high computational power, these smart vision chips include both image-acquisition and image-processing functions on a single chip. The *MIT* group developed both SIMD and MIMD array processors. The MIMD ones use mixed digital/analog arithmetic units to decrease the silicon area at the cost of the limited resolution. These array processors are now being integrated with image acquisition functions.

Real-time stereo vision system: the *MIT* group developed a cost-effective stereo vision system. Thanks to the use of a standard PC and two off-the-shelf plug-in boards, this system is able to run at the speed of 30 frames per second. The computational resource is focused on edge areas and sub-pixel operations are carried out in real-time. The system is used for three-dimensional lane detection and traffic monitoring, as well as, for other on-vehicle applications. The system is able to separate partially overlapped vehicles and distinguish them from shadows (see figure 3.9).



(a)



(b)

Fig. 3.9 (a) The selected edge pixels (marked using different grey-tones) and (b) their computed distance (published courtesy of Ichiro Masaki and Nicole Love – *Massachusetts Institute of Technology*).

3.2.12 Research carried out at the Phoang University of Science and Technology (Korea)

The *Phoang University of Science and Technology*, in cooperation with the *Hyundai* car company, developed three autonomous vehicles: the PRV series (POSTECH Road Vehicle) [83]. The newest of these systems, the PRV III (a Sonata III Hyundai car), is a fully autonomous vehicle able to perform road following, obstacle detection, and platooning. The PRV III vehicle is equipped with a color CCD camera used for Lane Detection, whilst Obstacle Detection and Platooning are performed by means of a laser range sensor and sonar sensors. The processing system is based on a PC-hosted TMS/C40-based parallel architecture; the vehicle also features

actuators for the steering wheel, brake, and throttle.

The whole system is made up of a modular structure in which a number of modules work according to their independent features and roles [84]; also the Lane Detection algorithm is based on the following different modules.

Edge module: this module relies on the presence of road markings and combines both neural networks and simple vision algorithms. Two parallel stripes of the acquired image are selected and filtered using Gaussian masks and zero crossing to find vertical edges. The result is matched against a given model (a typical road pattern) to compute a steering angle and a *fitness* evaluation indicating confidence in the result. The model is continuously updated taking into account the last result in order to reflect the current road scenario.

Color module: it is based on the processing of color images and is also intended to work on unstructured roads. When lane markings are present a procedure similar to the one described in the previous paragraph is used: the only difference is that the color of the markings is also determined for differentiating lateral or central markings. Conversely, when the road does not feature lane markings or in presence of strong shadows, a pyramidal approach that also works on unstructured roads is used [83].

This module too, outputs a steering angle and the fitness evaluation parameter.

Neural Steering Module: the output of the previously discussed modules is not directly fed to actuators: a module based on neural nets processes these outputs and drives the actuators mimicking human driving.

All the functionalities of the PRV III vehicle have been tested both on highways roads and on unpaved country roads.

This page is intentionally left blank

PART II

THE ARGO PROJECT

Downloaded from www.worldscientific.com
by 2601:282:700:6b90:5054:63cf:a8f2:87b6 on 12/22/24. Re-use and distribution is strictly not permitted, except for Open Access articles.

This page is intentionally left blank

This part of the book contains the presentation of the research work developed at the *Dipartimento di Ingegneria dell'Informazione, Università di Parma*, Italy. This research activity has been initially developed within the framework of the Eureka PROMETHEUS project, aimed at improving road traffic safety, and supported by the *Consiglio Nazionale delle Ricerche* under the frame of the *Progetto Finalizzato Trasporti II*.

Chapter 4 details the algorithms developed for both Lane and Obstacle Detection; chapter 5 presents the evolution of the computing architecture starting from the design of ad-hoc systems, up to the use of general-purpose off-the-shelf systems. Finally, chapter 6 describes the equipment of the ARGO vehicle.

This page is intentionally left blank

Chapter 4

Algorithms for Image Processing

This chapter presents the evolution of the algorithms developed at the *Università di Parma* for Lane Detection and Obstacle Detection. The first implementation of Lane Detection was based on a template-matching approach: a triangular template representing a perspective view of the road region was distorted and adapted to a subsampled copy of the acquired image with an iterative process.

Being based on information obtained from the luminance gradient of the incoming image, this approach was successful in normal conditions, but produced wrong results in presence of shadows or occlusions (e. g. produced by obstacles). The problem of shadows was solved thanks to the use of a specific filter based on the removal of the perspective effect, whilst obstacles were localized by another template-matching filter able to locate the ones with a rectangular shape.

The robustness of the perspective-based filter and limitation in detecting potential obstacles with a specific shape suggested shifting the approach towards a pattern-matching one, in which also the localization of obstacles is based on a perspective transform and does not suffer from shape limitations.

4.1 Lane Detection: a Model-Based Approach

This section presents an expectation-driven approach to low-level image segmentation aimed at road detection. This can be naturally mapped onto mesh-connected massively parallel SIMD architectures capable of handling hierarchical data structures. This approach represents a step towards the use of *top-down* control instead of the traditional *data-driven* approach,

which is generally used for data-parallel algorithms.

The input image is assumed to contain a distorted version of a given template; a multi-resolution stretching process is used to re-shape the original template in accordance with the acquired image content, minimizing a potential function. The distorted template is the process output.

The model which contains a-priori knowledge of the feature to be extracted (road or lane) is encoded in the traditional data structure handled in low-level processing: a two-dimensional array. In this case, a binary image has been chosen representing two different regions (road and off-road). From now on, this image will be referred to as *Synthetic Image*. It is obvious that different synthetic images must be used according to different acquisition conditions (camera position, orientation, optics, etc., which are fixed) and in accordance with the environment (number of lanes, one way or two way traffic, etc., which may change at run-time). A-priori world knowledge enables the correct selection of the synthetic model. As an example, figure 4.1 presents several different synthetic images for different conditions.



Fig. 4.1 Synthetic images used as road models for: (a) different camera positions and/or orientations (b) different number of lanes (assuming right-hand side driving).

This algorithm was implemented on the special-purpose massively parallel architecture called PAPRICA (described in section 5.1) and was tested on the MOB-LAB vehicle.

4.1.1 The multi-resolution approach

The image encoding the model (*Synthetic Image*) and the image coming from the camera (*Natural Image*) cannot be directly compared with local computations, because the latter contains much more details than the former. From all the known methods used for decreasing the presence of details, it is necessary to choose one that does not lower the strength of the feature to be extracted. For this purpose, a low-pass filter, such as

a 3×3 neighborhood-based anisotropic average filter, would not only reduce the presence of details, but also the sharpness of road boundaries, rendering their detection more difficult. Since the road boundaries exploit a long-distance correlation, a subsampling of both the natural and synthetic image would lead to a comparison less dependent on detail content. More generally, it is much easier to detect large objects at a low resolution, where only their main characteristics are present, than at a high resolution, where details of the specific represented object can make its detection more difficult. The complete recognition and description process, on the other hand, can only take place at high resolutions, where it is possible to detect even small details because of the preliminary results obtained at a coarse resolution.

These considerations lead to the use of a pyramidal data structure [5, 129, 143], made up of the same image at different resolutions. Many different architectures have been developed to support this computational paradigm [27–29, 60]: where the computing architecture contains a number of processing elements which is smaller than the number of image pixels and an external processor virtualization mechanism is used. A useful side effect caused by resolution reduction is a decrease in the number of computations to be performed. Therefore, the choice of a pyramidal computational paradigm, besides being supported by theory, offers an advantage in terms of computational efficiency.

4.1.2 The algorithm structure

As shown in figure 4.2, the natural image is filtered before each subsampling. In this way it is possible to decrease both the influence of noise and redundant details, and the distortion due to *aliasing* [120] introduced by the subsampling process. The image is partitioned into non-overlapping square subsets of 2×2 pixels each; the filter comprises a simple average of the pixels' values for a given subset, which reduces the signal bandwidth. The set of resulting values forms the subsampled image.

The stretching of the synthetic image is performed through an iterative algorithm (*Driven Binary Stretching*, DBS), a much simpler and morphological version of the *snake* technique [14, 36, 37, 82]. The result is then oversampled, and further improved using the same DBS algorithm, until the original resolution is reached. The boundary of the stretched template represents the final result of the process.

Downloaded from www.worldscientific.com
by 2601:282:700:6b90:5054:63cf:a8f2:87b6 on 12/22/24. Re-use and distribution is strictly not permitted, except for Open Access articles.

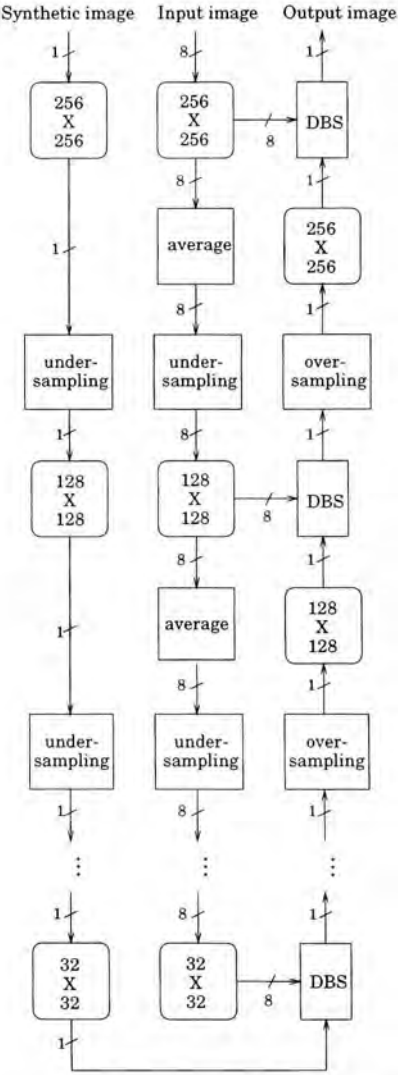


Fig. 4.2 Block diagram of the whole algorithm; for each step the depth of every image is shown (in bit/pixel).

4.1.2.1 The DBS filter

The purpose of the DBS filter, illustrated in figure 4.3, is to stretch the binary input model in accordance with the data encoded in the grey-tone natural image and to produce a re-shaped version of the synthetic model as an output.

Usually the border of a generic object is represented by brightness discontinuities in the image reproducing it; therefore, in the first version, the first step of the DBS filter includes an extremely fast and simple gradient-based filter computed on the 3×3 neighborhood of each pixel. Then, as shown in figure 4.3, a threshold is applied to the gradient image, in order to keep the most significant edges only. More precisely, two different threshold values are computed for the left and right halves of the image, in order to detect both road boundaries even under different conditions of light.

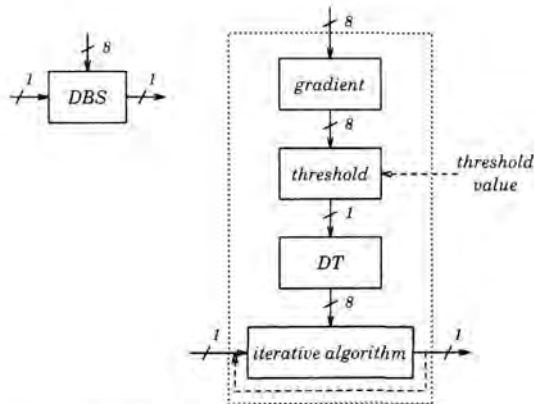


Fig. 4.3 Block diagram of the DBS filter.

Since a 2D mesh-connected massively parallel architecture was used, an iterative algorithm had to be performed in order to stretch the synthetic model towards the positions encoded in the thresholded image. A further advantage of the pyramidal approach is that the number of iterations required for successful stretching is low at a coarse resolution. This is because the image size is small; and again, only a few iterations are required for high-resolution refinement, due to the initial coarse low-resolution stretch-

ing. Each border pixel of the synthetic image is attracted towards the position of the nearest foreground pixel of the thresholded image, as shown in figure 4.4.

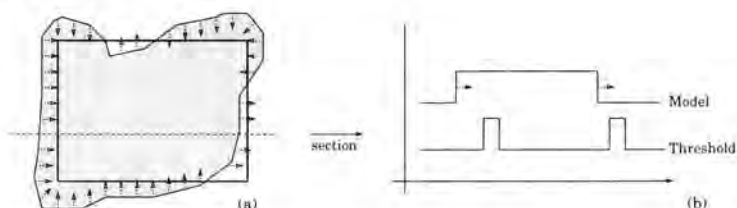


Fig. 4.4 The boundary pixels' attraction towards the thresholded image: (a) two-dimensional case with the model shown in grey and the thresholded image with a rectangular shape; (b) mono-dimensional section.

For this purpose, a scalar field V is defined on E^2 . Field $V : E^2 \rightarrow E$, links the position of each pixel $p \in E^2$ to a scalar value $V(p) \in E$, which represents the *potential* associated to the pixel itself. The set of these values is encoded in a *potential* image. The difference $V(p) - V(q)$ represents the *cost* of moving pixel p toward position q . The scalar field is defined in such a way that a negative cost corresponds to the movement towards the nearest position of the foreground pixels t_i belonging to the thresholded image. As a consequence, the iterative process is designed explicitly to permit all the pixel movements associated to a negative cost.

Therefore, the value $V(p)$ depends on the minimum distance between pixel p and pixels t_i :

$$V(p) = - \min_i d(p, t_i) , \quad (4.1)$$

where $d : E^2 \times E^2 \rightarrow E$ represents the distance between two pixels, measured with respect to a given metric. In this case, due to the particular simplicity of the implementation, a *city block* (*Manhattan distance*) metric has been chosen (see figure 4.5).

A very efficient method employed for the potential image computation using 2D mesh-connected architectures is based on the iterative application of morphological dilations:

- (1) a scalar counter is initialized to 0; for parallel architectures which cannot run any fragment of scalar code, this counter is associated to

4	3	2	3	4
3	2	1	2	3
2	1	0	1	2
3	2	1	2	3
4	3	2	3	4

Fig-4.5 City block or Manhattan distance from the central pixel.

every pixel in the image, therefore constituting a *parallel counter*;

- (2) the counter is decremented;
- (3) the binary input image is dilated using a 4-connected structuring element N , formed by the following elements:

$$N = \{(0, 1); (0, -1); (0, 0); (1, 0); (-1, 0)\} = \begin{array}{|c|c|c|} \hline & \bullet & \\ \hline \bullet & \bullet & \bullet \\ \hline & \bullet & \\ \hline \end{array} ; \quad (4.2)$$

- (4) the counter's value is assigned to the potential image in the positions where the pixel, due to the previous dilation, changes its state from *background* to *foreground*;
- (5) the process is then repeated from step 2, until the output of the morphological dilation is equal to its input.

This demonstrates that the potential image can be generated by the application of a *Distance Transform*, DT [15] to the binary thresholded image. Due to a more efficient implementation-dependent data handling, the final version of the potential image DT is obtained by adding a constant λ to every coefficient, so as to work with only positive values: λ represents the maximum value allowed for grey-tone images*. Thus the new definition of the scalar field V is:

$$V(p) = \lambda - \min_i d(p, t_i) . \quad (4.3)$$

Furthermore, since the distance information is only used by the pixels belonging to the synthetic image's border and their neighbors, the iterative process is stopped when the DT has been computed for these pixels, producing a noticeable performance improvement.

* In this specific case, since 8-bit images are considered, $\lambda = 255$.

As already mentioned, the core of the algorithm is an iterative process whose purpose is to move the synthetic image's edge pixels in the directions in which the DT gradient has a maximum. Figure 4.6 shows an example of a mono-dimensional stretching. As shown in appendix A, the strength of this approach lies in the fact that the stretching algorithm can be expressed simply by a sequence of morphological operations, and can therefore be mapped efficiently on mesh-connected massively parallel architectures, achieving higher performance levels than other approaches [36,37,82].

In order to describe the DBS algorithm, some definitions are introduced

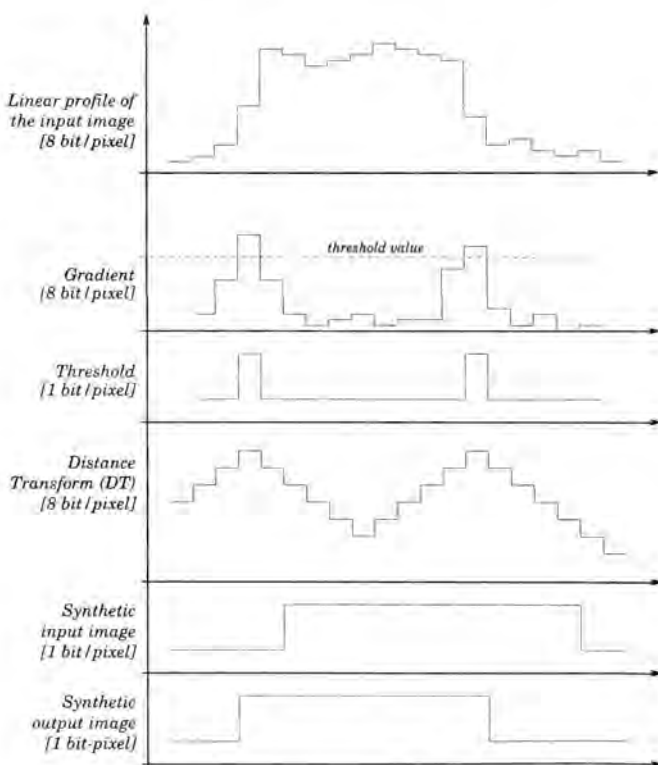


Fig. 4.6 Example of 4 iterations of the DBS algorithm (mono-dimensional case).

using Mathematical Morphology operators such as dilation ($\oplus$), erosion ($\ominus$), or complement ($\bar{\cdot}$). A two-dimensional binary image S is represented as a subset of E^2 , whose elements correspond to the *foreground* pixels of the image:

$$S = \left\{ s \in E^2 \mid s = (x, y), x, y \in E \right\}, \quad (4.4)$$

where vector (x, y) represents the coordinates of the generic element s .

The *external edge* of S is defined as the set of elements representing the difference between S and the dilation of S by the 4-connected structuring element N shown in expression (4.2):

$$B_e(S) = (S \oplus N) \cap \bar{S}. \quad (4.5)$$

In a similar way, the set of elements representing the difference between S and its erosion using the same structuring element N is defined as the *internal edge* of S :

$$B_i(S) = \overline{(S \ominus N)} \cap S. \quad (4.6)$$

The Iterative Rules

The set of elements $B(S) = B_e(S) \cup B_i(S)$ are the only elements which can be inserted or removed from set S by the application of a single iteration of the DBS algorithm. More precisely, two different rules are applied to the two edges: the first, applied to the external edge $B_e(S)$, determines the elements to be included in set S ; while the second, applied to the internal edge $B_i(S)$, determines the elements to be removed from set S .

(1) Rule for the external edge:

- each pixel of the external edge of S computes the minimum value of the DT associated to its 4-connected neighbors belonging to set S ;
- all the pixels, whose associated DT is greater than the value previously computed, are inserted into set S .

The application of this rule has the effect of expanding the synthetic image towards the foreground pixels t_i which are not included in the synthetic model (see the right hand side of figure 4.4.b).

(2) Rule for the internal edge:

- each pixel of the internal edge of S computes the minimum value of the DT associated to its 4-connected neighbors not belonging to set S ;
- all the pixels, whose associated DT is greater than the value previously computed, are removed from set S .

The application of this rule has the effect of shrinking the synthetic image (see the left hand side of figure 4.4.b).

Note that rule 2 is the inverse of rule 1: the latter tends to stretch the foreground onto the background, while the former acts in the opposite way, using the complement of the synthetic image.

Flat Handling

Figure 4.6 refers to a mono-dimensional stretching. Unfortunately, when dealing with 2D data structures, the DT image does not present a strictly increasing or decreasing behavior, even locally. Therefore, an extension to the previous rules must be considered for correct flat-handling in the 2D space.

Figure 4.7.a shows the modulus of the DT coefficients (in the case of $\lambda = 0$), together with the input binary image (shown in grey). The output of the iterative DBS algorithm is presented in figure 4.7.b.

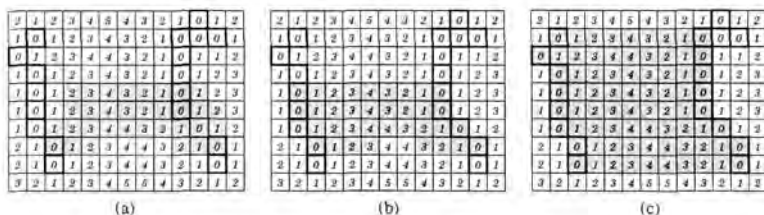


Fig. 4.7 Two-dimensional stretching: (a) input image; (b) stretching without flat-handling; (c) stretching with flat-handling. Different square markings represent different states of input binary image; dark grey areas represent the stretching area.

Since the movement of a generic pixel towards positions holding an equal DT coefficient is expressly disabled, the resulting binary image does not completely follow the shape encoded in the DT image. A minor revision

of the definition of rule 1 is therefore required. Figure 4.7.c is obtained with the following rule applied to the external edge.

- Rule for the external edge, including flat handling:
 - each pixel of the external edge of S computes the minimum value of the DT associated to its 4-connected neighbors belonging to set S ;
 - all the pixels, whose associated DT is greater than the value previously computed, and all the pixels not belonging to the thresholded image, whose associated DT is equal to the value previously computed, are inserted into set S .

The specific requirement for the pixels moving towards a flat region, of not belonging to the thresholded image, ensures that the binary image does not follow the DT chain of maxima. With such a requirement, in the specific case of figure 4.7, the maxima in the upper-right hand area are not included in the resulting binary image.

4.1.3 Performance analysis

The algorithm discussed in this section maps naturally onto a single-bit mesh-connected SIMD machine, because the whole computation can be efficiently expressed as a sequence of binary morphological operators, as shown in appendix A.

The complete processing was first tested on a Connection Machine CM-2 [72] and then implemented on the special-purpose massively parallel SIMD architecture PAPRICA installed on the MOB-LAB vehicle. The specific processor virtualization mechanism utilized by the PAPRICA architecture allows the handling of pyramidal data structures without any additional overhead equipment.

Table 4.1 presents the computational time required by each step of the algorithm (considering 5 DT and 5 DBS iterations for each pyramid level). The complete processing of a single frame ($256^2 \rightarrow 32^2 \rightarrow 256^2$) takes a little more than 500 ms. Considering the image acquisition and the data transfer time, the system allows a processing rate of 1 Hz.

Due to the high correlation between two consecutive sequence frames, the final stretched template can be used as the input model to be stretched by the processing of the following frame. In these conditions a lower num-

Table 4.1 Performance of the PAPRICA system; the numbers refer to 5 iterations of the DT process and 5 iterations of the DBS filter for each pyramid level.

Operation	Image size	Time [ms]
Resolution Reduction	$256^2 \rightarrow 32^2$	149.70
DT, DBS & Oversampling	$32^2 \rightarrow 64^2$	18.55
DT, DBS & Oversampling	$64^2 \rightarrow 128^2$	69.13
DT, DBS & Oversampling	$128^2 \rightarrow 256^2$	296.87
Total	$256^2 \rightarrow 32^2 \rightarrow 256^2$	534.25

ber of DBS and DT iterations are needed for complete template reshaping, consequently producing a noticeable performance improvement. The reduction in the time required to process a single frame also increases the correlation between the current and the following frame in the sequence, and as a result allows a further reduction in computation time[†].

Images acquired in many different conditions and environments have been used for extensive experimentation, which was first performed off-line on a functional simulator implemented on an 8k processor Connection Machine CM-2 and then in real-time on the PAPRICA hardware itself on MOB-LAB. The complete system was demonstrated at the final PROMETHEUS project meeting in Paris, in October 1994: the MOB-LAB land vehicle was driven around the two-lane track at Morte-fontaine, under different conditions (straight and curved roads, with shadows and changing light conditions and with other vehicles on the path).

This demonstration made possible the detection of the system's main limitations and enabled a critical analysis of the approach, consequently leading to proposals for its modifications.

4.1.4 Critical analysis and evolution

The discussed approach permits the achievement of a remarkable performance in terms of both computation time (when mapped on massively parallel architectures) and output quality. The performance achieved in

[†] During the processing of highly correlated sequences (namely when the road conditions are subject to a slow change or in an off-line slow-motion tape playback in laboratory) the computational time required by the processing of a single frame can reach a minimum value of 150 ms.

terms of output quality are good when the model matches (or is sufficiently similar to) the road conditions, namely when road markings are painted on the road surface, inducing a sufficiently high luminance gradient (see figures 4.8 and 4.9 which represent a segment of a straight road and a segment of a curved road). This is a successful approach when the road or lane boundaries can be extracted from the input image through a gradient thresholding operation (see figures 4.8.b and 4.9.b). Unfortunately, this is not always possible, for example when the road region is a patch of shadow or sun spots, as in figure 4.10.a. In this case the computation of the DT starting from the thresholded image (shown in figure 4.10.b) is no longer significant and the road region cannot be successfully recovered as shown in figure 4.10.c and 4.10.d. A different method must be devised for the computation of the binary image to be used as input for the Distance Transform.

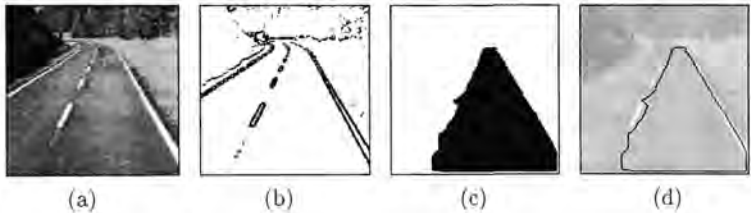


Fig. 4.8 Lane Detection on a straight road: (a) input image; (b) image obtained by thresholding the gradient image; (c) stretched template; (d) superimposition of the stretched template's edges onto the original image.

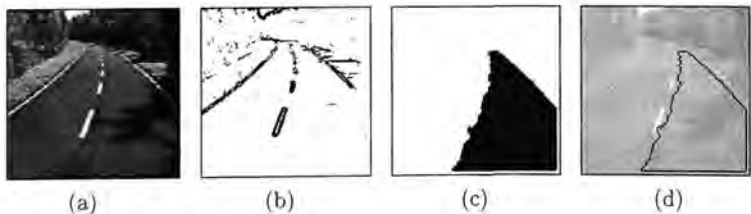


Fig. 4.9 Lane Detection on a curved road: (a) input image; (b) image obtained by thresholding the gradient image; (c) stretched template; (d) superimposition of the stretched template's edges onto the original image.

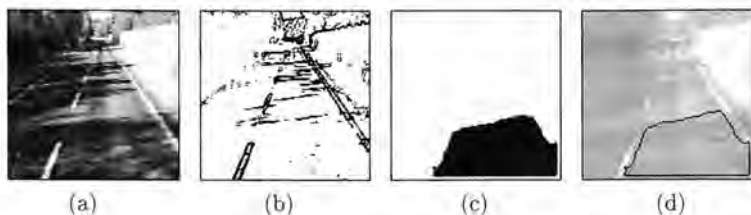


Fig. 4.10 Lane Detection on a straight road with shadows: (a) input image; (b) image obtained by thresholding the gradient image; (c) stretched template; (d) superimposition of the stretched template's edges onto the original image. This is a case where the thresholded gradient (b) cannot be used to determine the DT image due to the noise caused by shadows.

An approach based on the removal of the perspective, deeply described in section 4.3.1, has been introduced to improve the system performance in shadow conditions. A non-uniform resampling is applied to the input image (figures 4.11.a); assuming a flat road, every pixel of the resampled image (figures 4.11.b) represents the same portion of the road. Due to their constant width within the overall image, the road markings can now be easily enhanced by extremely simple morphological filters (figures 4.11.c). Finally the perspective effect is reintroduced (figures 4.11.d). Figures 4.11.e show the superimposition of figures 4.11.d onto brighter versions of the original ones.

The replacement of the gradient thresholding with the perspective-based filter proved to be highly effective. This *extended* version of the DBS filter is shown in figure 4.12. However, even if the perspective-based filter alone is able to extract the road markings with a high degree of confidence, the hierarchical application of the DBS filter (and its extension to the handling of image sequences) is of basic importance, since it allows exploitation of the temporal correlation between successive sequence frames (solution tracking).

Since the removal (and re-introduction) of the perspective effect can be reduced to a mere image resampling and the filter is based on simple morphological operators, implementing on the PAPRICA system is straightforward. The results obtained are good both for the output quality (the problems caused by shadows are resolved) and for computational time: a single frame is processed in less than 100 ms, therefore allowing the processing of

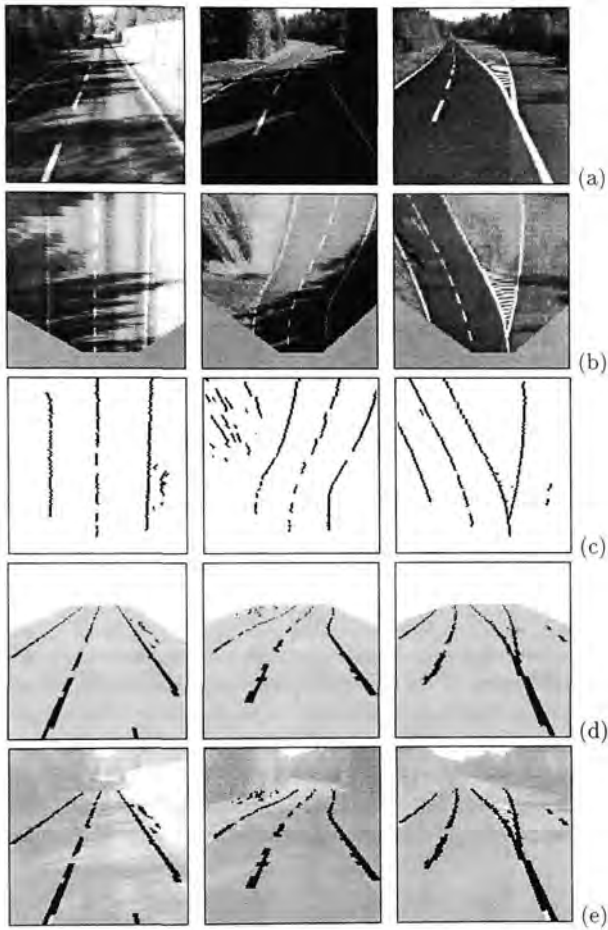


Fig. 4.11 The detection of road markings through the removal of the perspective effect in three different conditions: straight road with shadows, curved road with shadows, junction. (a) input image; (b) re-sampled image; (c) result of the line-wise detection of black-white-black transitions in the horizontal direction; (d) reintroduction of the perspective effect, where the grey areas represent the portion of the image shown in (c); (e) superimposition of (d) onto a brighter version of the original image (a).

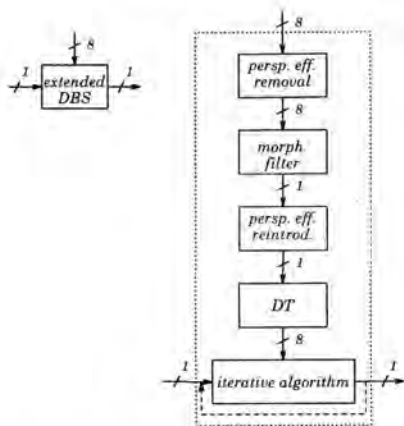


Fig. 4.12 Diagram of the extended DBS filter including the perspective-based filtering.

about 10 frames per second. The improvement of the DBS process by means of the perspective-based filter, in addition to allowing correct road (or lane) detection in the presence of shadows, can be implemented extremely efficiently on the PAPRICA system, taking advantage of a specific hardware extension designed explicitly for this purpose (non uniform-resampling).

The general nature of the presented approach enables the detection of other sufficiently large-sized features: for example, using different synthetic models it is possible to detect road or lane boundaries, as shown in figure 4.13.

A problem that still remains to be addressed is shown in figure 4.14, where the lane cannot be successfully detected due to the presence of an obstacle.

4.2 Obstacle Detection: a Model-Based Approach

This section contains the presentation of the vehicle detection system acting on monocular video sequences which was initially implemented on the MOB-LAB land vehicle.

The hardware platform is composed of a pipeline of two different computing engines: PAPRICA and a traditional serial architecture running

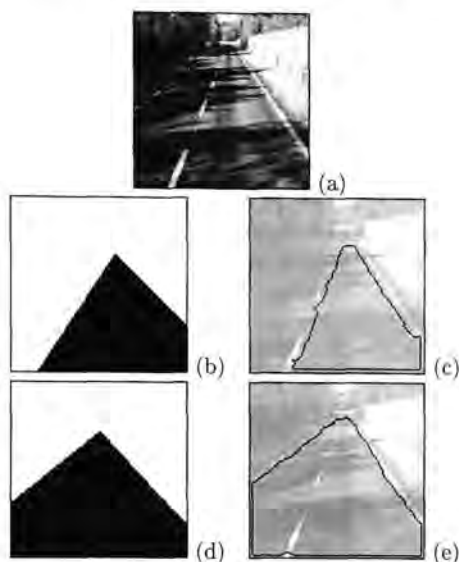


Fig. 4.13 Use of different templates: (a) input image; (b) synthetic model used for lane detection; (c) superimposition of edges belonging to the stretched model onto the original input image; (d) synthetic model used for road detection; (e) superimposition of the stretched model's edge onto the original input image.

medium-level tasks aimed at detecting the vehicles' position in the sequence.

Since a generic obstacle (vehicle) possesses an almost rectangular shape, the system's aim is to extract from the incoming image all the possible rectangles that may represent vehicles on the path.

4.2.1 The vehicle detection algorithm

The preliminary image filterings and the detection of the features of interest (corners of the rectangles) are performed by the massively parallel system PAPRICA, by means of a mathematical morphology-based computational paradigm. Conversely, the following selection of the meaningful corners, the grouping of 4 corners representing a rectangle (or trapezoid), and the final validation of the determined polygons are performed on the serial system. Due to its simplicity, the following high-level interpretation of the results is performed by the same sequential system.

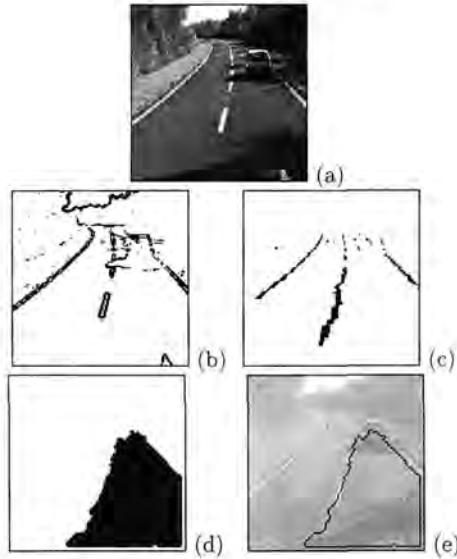


Fig. 4.14 Lane Detection on a straight road with a vehicle in the path: (a) input image; (b) gradient image; (c) result of the removal of the perspective effect, morphological filtering, and reintroduction of the perspective effect; (d) stretched template; (e) superimposition of the stretched model's edges onto the original input image. This is the only case in which the DBS algorithm cannot stretch the binary model successfully, due to the presence of a vehicle occluding the road markings.

4.2.1.1 Low-level processing

The incoming image (a single field, 512×256 pixels) is acquired and loaded directly into PAPRICA's image memory. As shown in section 5.1, the PAPRICA system provides a mechanism for the emulation of pyramidal interconnections between processors without any computational overhead: the image is in fact undersampled to obtain a working resolution of 256×256 pixels. Then, an average operation on a 3×3 neighborhood is performed prior to the gradient filter, in order to diminish the influence of noisy spikes. Then, a threshold is applied for the binarization of the resulting image: in this case, due to the specific subsequent processing, an accurate determination of the threshold is not necessary. Under these circumstances, a low threshold is used in order to keep a high amount of edges; the edges which

do not correspond to rectangles will be deleted by the following medium-level step.

Through this choice, the binarized image consists of wide edges, but a thinning filter [128] is not recommended. In fact, although the lines' width is reduced, therefore leading to a smaller number of black pixels (and consequently to a faster subsequent processing), it does not permit the detection of corners with a simple and traditional pattern-matching technique. Figure 4.15.b shows the binarized version of figure 4.15.a. The low-level processing ends with the determination of 4 binary images indicating the presence of four kinds of corners: figure 4.16 shows the four patterns used, which are not symmetrical; in fact, experimental tests have demonstrated that longer horizontal lines also allow the detection of imperfect rectangles (trapezoids). Tests on different road conditions have shown that the horizontal lines are more important and less noisy than vertical ones.

4.2.1.2 Medium-level processing

In this section obstacles' edges are presumed to be rectangles. The main goals of the medium-level computation are the following.

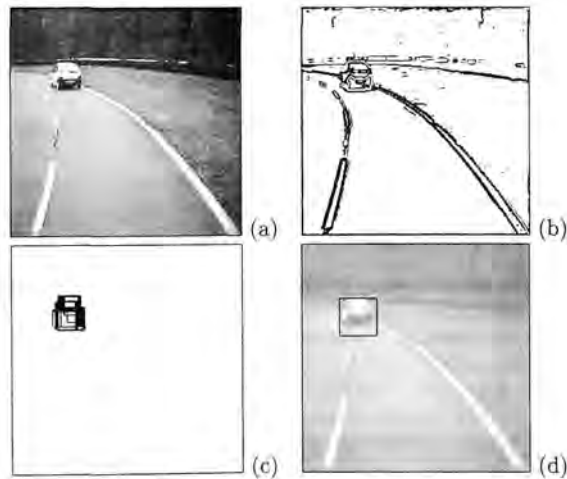


Fig. 4.15 Shape-based Obstacle Detection: (a) original image, (b) binarized image, (c) rectangles detection, (d) final result.

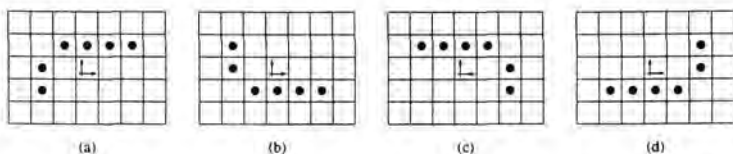


Fig. 4.16 Patterns used to detect the 4 corners of a rectangle.

Detection of rectangles

The algorithm scans the whole image looking for upper-left corners (figure 4.16.a). For each detected corner, it looks for a matching lower-left one (figure 4.16.b), then a lower-right (figure 4.16.d) and finally an upper-right one (figure 4.16.c). It is possible to avoid many computations reducing the number of rectangles before their complete construction: the side length can only range in a precise interval. This interval depends on the obstacle's position within the image: due to the perspective effect induced by specific acquisition conditions, rectangles detected in the lower part of the image correspond to vehicles close to the camera. For this reason, their size should be larger than the size of similar objects detected in the upper part of the image. Moreover, real obstacles are characterized by a specific aspect-ratio interval: this permits the rejection of a large amount of rectangles, providing a speed-up in the following steps of the processing. The rectangle construction procedure is drafted in the pseudo-code as follows:

```

for every upper left angle
  for every matching lower left angle
    lx = vertical side length
    if min_dim < lx < max_dim
      for every matching lower right angle
        ly = horizontal side length
        if (min_dim < ly < max_dim) and
           (min_ratio < lx/ly < max_ratio)
          for every matching upper right angle
            if (upper right angle) matches (upper left angle)
              then a rectangle is detected
          end for
        end for
      end for
    end for
  end for
end for

```

Rectangles validation

The validation procedure consists of checking, in the binarized image, for the presence of over-threshold pixels in the positions corresponding to the rectangle's sides. A side is considered valid if and only if at least half of the pixels representing it are actually present in the binarized image. Rectangles with at least one invalid side are rejected.

Obviously, in order to speed-up the computation, the validation procedure is interleaved with the rectangle construction process: conversely, for the sake of the presentation's simplicity, the two steps have been discussed separately as follows.

Fusion step

Normally, for each obstacle the algorithm detects more than one rectangle, each one with small differences in position and size, as shown in figure 4.15.c. Although computational time is increased (a larger number of rectangles must be detected and validated), this redundancy offers the advantage of reducing the risk of missing obstacles. Therefore, a fusion algorithm is used to reduce the amount of redundant information, simplifying the following interpretation of the results. If two or more rectangles are sufficiently overlapped, they are replaced by a larger one, as in figure 4.15.d.

The fusion step is drafted in the following pseudo-code:

```

for r1 = every valid rectangle
  for r2 = every different valid rectangle
    if overlap_ratio(r1, r2) > max_overlap_ratio
      new_rectangle->position =
        weighted_average(r1->position, r2->position)
      if (r2 is included in r1)
        new_rectangle->dim = r2->dim
      else
        new_rectangle->dim = r1->dim + r2->dim + distance(r1,r2)
        mark r1 and r2 as not valid
      end for
    end for
  end for

```

The overlap ratio of rectangle $r1$ over $r2$ is defined as the ratio between the area of the intersection between $r1$ and $r2$, and the area of $r1$. The weighted average is computed using the size of rectangles as weight. Since generally all the rectangles are heavily overlapped, the fusion step is reduced to the growing of a single rectangle, including all the others.

The algorithm has also been extended to handle image sequences. In this case, a notable speed-up is achieved by dynamically reducing the area of interest [107, 159], namely the region where the obstacle is supposed to be in the following frame of the sequence. It is reduced to a single image window where the obstacles have been previously detected: these windowing techniques [67, 151] allow a fast tracking of the moving obstacle. The region of interest is determined by taking the smaller bounding box including all the obstacles. Information on vehicle motion and the steering wheel angle can be used to update the window of attention in a more accurate fashion. However, periodically or when no obstacles are found, the entire image is analyzed, in order to avoid the risk of missing new obstacles.

4.2.2 Performance analysis

The whole process has been tested on images representing several extra-urban double-lane roads, with painted road markings. Although the specific process does not rely on the presence of road markings, it has been tested on structured roads since it had to be integrated with the module aimed at the identification of lane boundaries, and therefore based on road markings detection. Unfortunately, road markings represent a noisy feature that slows down the medium-level phase: in fact, the number of detected corners is increased as well as the number of required matchings. Nevertheless, generally the rectangles generated by successful matching do not comply with the size requirement mentioned in section 4.2.1.2. For this reason, the final amount of valid rectangles is not affected by the presence of road markings, although sensibly slackened.

Figure 4.17 presents a few of the results regarding the processing of images that show a vehicle on the road: the resolution of the incoming grey-level images is 256×256 , 8 bit/pixel.

The algorithm has been tested in different conditions: without obstacles, on straight and curved roads, in different light conditions, and on junctions. The more critical behavior corresponds to the presence of shadows on the path, inducing the appearance of a considerable number of edges in the binarized image.

Table 4.2 presents the timings of low- and medium-level processings: the former does not depend on the specific image, while the latter is strongly influenced by the amount of extracted features (corners). Table 4.2 confirms

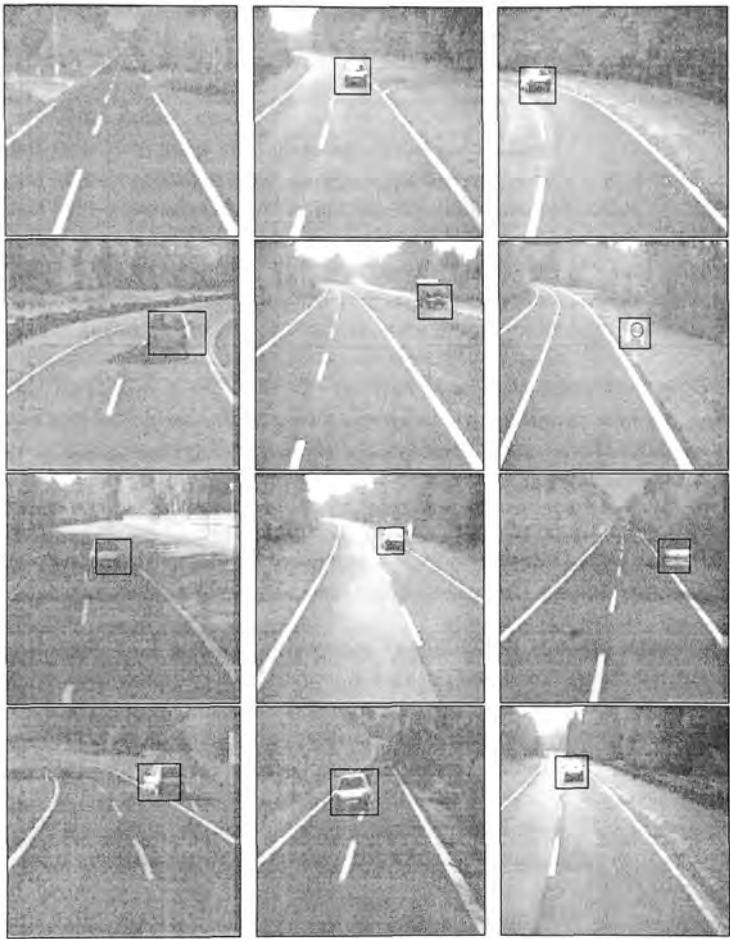


Fig. 4.17 Some examples showing the detection of obstacles with a rectangular shape.

that on straight and curved roads without vehicles the algorithm runs faster than in the same conditions with vehicles on the path. This is due to the overhead caused by the rectangles' validation phase. On the other hand, on particular kinds of images with a lot of details (edges) the number of

Table 4.2 Timings for vehicle detection in different conditions.

Operation	Road conditions	Time	Speed
Low level	-	265 ms/frame	3.77 Hz
Medium level	Straight road with one car	1.21 s/frame	0.82 Hz
	Straight road without cars	0.63 s/frame	1.58 Hz
	Curve road with one car	1.97 s/frame	0.50 Hz
	Curve road without cars	0.58 s/frame	1.74 Hz
	Junction with cars	1.63 s/frame	0.61 Hz
	Junction without cars	2.33 s/frame	0.43 Hz
	Mixed route with one car	1.38 s/frame	0.72 Hz

rectangles grow, consequently requiring a longer processing. In this case, the presence of one vehicle decreases the computational time thanks to the windowing techniques.

4.3 The GOLD System: a Cooperative Pattern-Matching Approach to Lane Detection and Obstacle Detection

This section presents the GOLD (Generic Obstacle and Lane Detection) system, a stereo vision-based software system to be used on moving vehicles to detect both generic obstacles (without constraints on symmetry or shape) and lane position in a structured environment (with painted lane markings). Thanks to the Inverse Perspective Mapping geometrical transform, the perspective effect is removed from both left and right stereo images; a single image is used to detect lane markings with a series of morphological filters, while both stereo images are used for the detection of free-space in front of the vehicle.

The system was initially tested on the MOB-LAB mobile laboratory which was driven for more than 3000 km along extra-urban roads and free-ways at speeds of up to 80 km/h. It demonstrated its robustness with respect to shadows and changing light conditions, different road textures, and vehicle movements. These extensive tests made it possible to improve and refine the system and, subsequently, use an enhanced version of the GOLD system as the automatic driving system on the ARGO autonomous vehicle (described in chapter 6).

4.3.1 Inverse Perspective Mapping

This section discusses a geometrical transform that permits the removal of the perspective effect from images.

In early vision, images can be considered as a mere collection of pixels; for this reason in the case of generic filterings, such as contrast enhancement or brightness rescaling, namely when the meaning of each pixel is not taken into account, the processing can be efficiently performed on SIMD architectures. Unfortunately, in almost all real cases, every image pixel carries its own amount of information, and therefore possesses a different meaning compared to other pixels: both the angle of view from which the scene is acquired and distance of the objects from the camera, contribute to associate a different information content to each pixel. The perspective effect must in fact be taken into account in order to weigh each pixel according to its information content, which is generally not uniformly and homogeneously distributed. The different functions applied to each pixel, in accordance to its position within the image, make the implementation of this kind of low-level processing inefficient on an SIMD system, which by definition performs the *same* operation on each of the image's pixel. As an example, let us consider the specific problem of road markings detection in images acquired from a camera installed on a mobile vehicle. Due to the perspective effect introduced by the acquisition conditions, the road markings width changes according to their distance from the camera (as shown in figure 4.18.a). Thus a pixel in the middle of this image represents

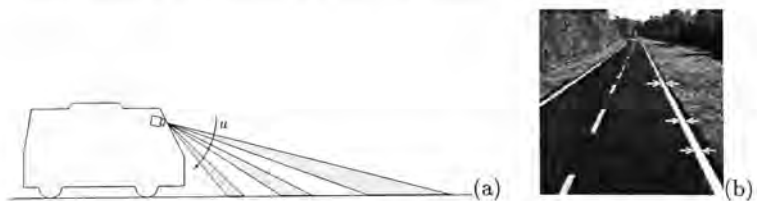


Fig. 4.18 (a) Due to the perspective effect different pixels represent different portions of the road; (b) the road markings width changes according to their position within the image.

tens or more cm^2 of the road texture, while a pixel in the lower part of the same image represents only a few cm^2 . Therefore, the correct detection of road markings should be based on matchings with different sized patterns,

according to the specific position within the image. Unfortunately this differentiated low-level processing cannot be efficiently performed on SIMD massively parallel systems.

To cope with this problem the Inverse Perspective Mapping (IPM) geometrical transform has been introduced: it enables the removal of the perspective effect from the acquired image remapping it into a new 2D domain (the *remapped domain*), in which the information content is uniformly distributed amongst all pixels, therefore allowing the efficient implementation of the following processing steps on an SIMD machine. Obviously, the application of the IPM transform requires the knowledge of the specific acquisition conditions (camera position, orientation, optics...) and some assumptions on the scene represented in the image (here defined as a-priori knowledge). For this reason the IPM transform can be of use in structured environments [117, 131], where, for example, the camera is mounted in a fixed position (e.g. for surveillance, quality control, or industrial inspection), or in situations where the system's calibration can be computed via other kinds of sensor, such as sonars, lasers, or bumpers (e.g. for indoor robotics applications [94, 140]).

4.3.1.1 Removing the perspective effect

The procedure aimed at removing the perspective effect resamples the incoming image, remapping each pixel towards a different position and producing a new two-dimensional array of pixels.

Assuming that the vision system acquires an image of a scene representing a known surface, the IPM transform removes the perspective effect and produces a remapped image that represents the stretched texture of the framed surface. In the following section, the discussion will be limited to the case of a planar surface $\mathcal{S}$, in this case the use of IPM makes it possible to obtain a bird's eye view of the scene.

The acquisition process can be devised as a transform from the 3D Euclidean space $\mathcal{W}$ to the 2D Euclidean space $\mathcal{I}$, where:

- $\mathcal{W} = \{(x, y, z)\} \in E^3$ represents the 3D world space (*world-coordinate system*), where the real world is defined;
- $\mathcal{I} = \{(u, v)\} \in E^2$ represents the 2D image space (*screen-coordinate system*), where the 3D scene is projected.

The $\mathcal{I}$ space represents the acquired image while, according to the flatness

assumption, the remapped image is defined as the xy plane of the W space, namely the $S = \{(x, y, 0) \in W\}$ surface.

The use of the IPM transform allows the projection the acquired image onto S , acting as the dual of a *ray-tracing* algorithm [109]. Figure 4.19 shows the relationships between the two spaces W and I .

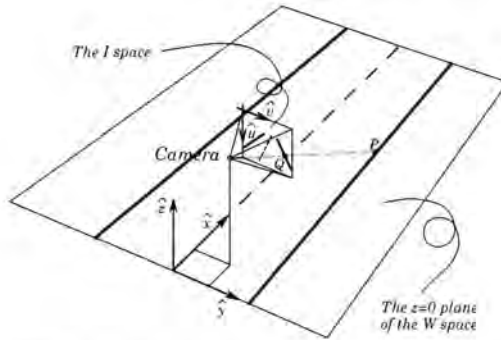


Fig. 4.19 The relationship between the two coordinate systems.

$I \rightarrow S$ mapping

The use of the IPM transform requires the knowledge of the following parameters [109]:

- *viewpoint*: the camera position is $C = (l, d, h) \in W$;
- *viewing direction*: the optical axis $\vec{o}$ is determined by the following angles:
 - $\bar{\gamma}$: the angle formed by the projection (defined by versor $\hat{\eta}$) of the optical axis $\vec{o}$ on the plane $z = 0$ and the x axis (as shown in figure 4.20.a);
 - $\bar{\theta}$: the angle formed by the optical axis $\vec{o}$ and versor $\hat{\eta}$ (as shown in figure 4.20.b);
- *aperture*: the camera angular aperture is 2α ;
- *resolution*: the camera resolution is $n \times n$.

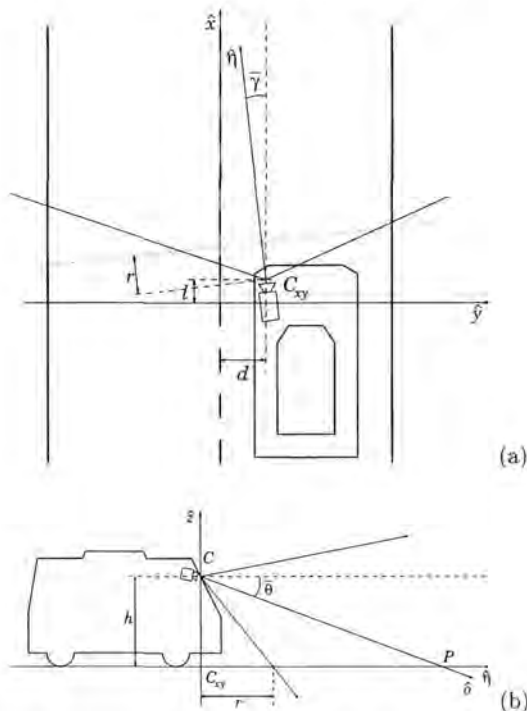


Fig. 4.20 (a) The xy plane in the $\mathcal{W}$ space, namely the S surface; and (b) the zn plane, assuming the Origin translated onto the projection C_{xy} of C on S .

In the following the intersection is determined between the

$$S = \{(x, y, 0) \in \mathcal{W}\} \quad (4.7)$$

surface and each of the n^2 rays that leave the camera (within the viewing pyramid [109]). The n^2 rays are determined by $\theta(u)$ and $\gamma(v)$, obtained by scanning the camera angular aperture. $\theta(u)$ and $\gamma(v)$ can vary as follows:

$$\begin{cases} \theta(u) = (\bar{\theta} - \alpha) + u \frac{2\alpha}{n-1} \\ \gamma(v) = (\bar{\gamma} - \alpha) + v \frac{2\alpha}{n-1} \end{cases} \quad (4.8)$$

where $u, v = 0, 1, 2, \dots, n-1$ represent the coordinates in the $\mathcal{I}$ space.

The distance d between the projection C_{xy} of $C \in \mathcal{W}$ on the $\mathcal{S}$ plane and the intersection $P \in \mathcal{S}$ between the generic ray determined by the couple $(\theta(u), \gamma(v))$ and $\mathcal{S}$ is given by the following expression:

$$d(C_{xy}, P) = \frac{h}{\tan[\theta(u)]} . \quad (4.9)$$

The projection of P on the x and y axes is given by:

$$\begin{cases} x(P) = \frac{h}{\tan[\theta(u)]} \cos \gamma(v) + l \\ y(P) = \frac{h}{\tan[\theta(u)]} \sin \gamma(v) + d \end{cases} . \quad (4.10)$$

Using equations (4.8), the final mapping $f: \mathcal{I} \rightarrow \mathcal{S}$ as a function of u and v is then given by:

$$\begin{cases} x(u, v) = h \times \cot \left[(\bar{\theta} - \alpha) + u \frac{2\alpha}{n-1} \right] \times \cos \left[(\bar{\gamma} - \alpha) + v \frac{2\alpha}{n-1} \right] + l \\ y(u, v) = h \times \cot \left[(\bar{\theta} - \alpha) + u \frac{2\alpha}{n-1} \right] \times \sin \left[(\bar{\gamma} - \alpha) + v \frac{2\alpha}{n-1} \right] + d \\ z(u, v) = 0 \end{cases} . \quad (4.11)$$

Equations (4.11) return the point $(x, y, 0) \in \mathcal{S}$ corresponding to point $(u, v) \in \mathcal{I}$.

$\mathcal{S} \rightarrow \mathcal{I}$ mapping

Following similar algebraic and trigonometric manipulations, the inverse transform $g: \mathcal{S} \rightarrow \mathcal{I}$ (the dual mapping) is given as follows. Given a generic point $P \equiv (x, y, 0)$, the angles are computed formed by the optical axis $\hat{o}$ and segment $\overline{CP}$. Following the definitions of θ and γ given by equations (4.8), it is:

$$\begin{cases} \gamma(x, y, 0) = \arctan \left[\frac{y-d}{x-l} \right] \\ \theta(x, y, 0) = \arctan \left[\frac{h \sin \gamma(x, y, 0)}{y-d} \right] \end{cases} . \quad (4.12)$$

If $\gamma(x, y, 0) \in [\bar{\gamma} - \alpha, \bar{\gamma} + \alpha]$ and $\theta(x, y, 0) \in [\bar{\theta} - \alpha, \bar{\theta} + \alpha]$, then point P belongs to the viewing pyramid and its coordinates (u, v) in the $\mathcal{I}$ space

are given by the following mapping $g: \mathcal{S} \rightarrow \mathcal{I}$:

$$\begin{cases} u(x, y, 0) = \frac{\theta(x, y, 0) - (\bar{\theta} - \alpha)}{\frac{2\alpha}{n-1}} \\ v(x, y, 0) = \frac{\gamma(x, y, 0) - (\bar{\gamma} - \alpha)}{\frac{2\alpha}{n-1}} \end{cases} \quad (4.13)$$

The use of the projection transform defined by equations (4.13) permits the removal of the perspective effect and recovery of the texture of the $\mathcal{S}$ surface (the $z = 0$ plane of the $\mathcal{W}$ space). The array of pixels of coordinates $(x, y, 0) \in \mathcal{W}$ (which form the remapped image) is scanned and each pixel is assigned the value of the corresponding pixel of coordinates $(u(x, y, 0), v(x, y, 0)) \in \mathcal{I}$.

As an example, figure 4.21 shows the original and remapped images. The advantage offered by the use of the IPM is that in the remapped image (see figure 4.21.c) the road markings width is almost invariant within the whole image. This simplifies the following detection steps and allows their implementation with a traditional pattern-matching technique on an SIMD system. Since a road detection algorithm needs to be fast, the projected image's resolution has been chosen as a trade-off between processing time and information loss; figure 4.21.d shows the remapped image, presenting the road surface with an aspect ratio of 1:1. Note that the specific camera position, orientation, and angular aperture cause the lower portion of the remapped image to be undefined; the size of the undefined area depends on the remapping parameters and on the acquisition setup.

4.3.2 Inverse Perspective Mapping and stereo vision

This section discusses an extension of the Inverse Perspective Mapping transform to stereo vision.

As a consequence of depth loss caused by the acquisition process, the use of a single two-dimensional image does not allow a three dimensional reconstruction of the world without the use of any a-priori knowledge. On the other hand, when the target is the reconstruction of 3D space, the solution gets more and more complex due to the larger amount of computation required by well-known approaches, such as:

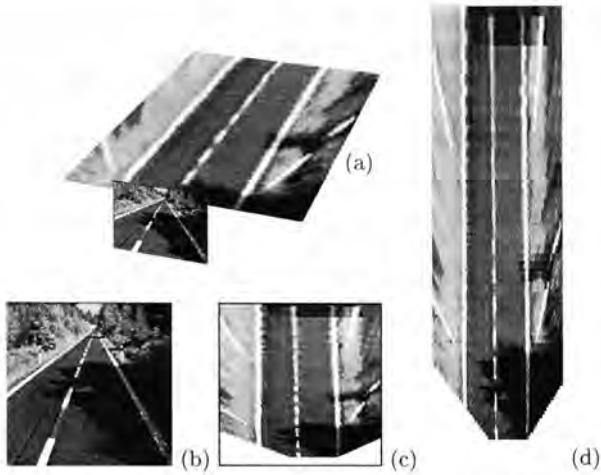


Fig. 4.21 IPM applied to a road environment: (a) 3D representation of the environment, (b) the acquired image, (c) the remapped image, and (d) the remapped image with the original aspect ratio.

- the processing of *stereo images* [58],
- the processing of *sequences of images* [69], or
- the use of other non-trivial techniques, such as, for example, *shape from shading* [111, 114] or *shape from texture* [142].

Using pairs of stereo images has generally been considered as a solution to this problem; the *traditional* approach to stereo vision [58] can be divided into four steps:

- (1) *calibration* of the vision system;
- (2) *localization* of a *feature* in an image;
- (3) *identification* and *localization* of the *same* feature in the other image;
- (4) *3D reconstruction* of the scene.

Triangulations between points that correspond to the same feature (*homologous points*) allow the solution to the problem of three dimensional reconstruction. Unfortunately, the determination of homologous points is a difficult task, but the introduction of some domain specific constraints

(such as the assumption of a flat road in front of the cameras) can simplify it. In particular, when a complete 3D reconstruction is not required and a *verification* of matching with a given model is sufficient, the application of IPM to stereo images plays a strategic role. For example the 3D reconstruction can be replaced with a model match in many cases, such as:

- *obstacle detection*: the model represents the environment without obstacles; any deviation from the model detects a potential obstacle;
- *object identification*: the model encodes the shape of a specific object; matching enables the recognition of the object by means of its shape;
- *object localization*: the model encodes the position of a specific object with respect to the environment; the measurement of the deviation from the model permits the determination of the actual object position.

The set of points $\mathcal{H} = \{(x, y, z) \mid u^{(L)} = u^{(R)} \text{ and } v^{(L)} = v^{(R)}\}$, where $(u^{(L)}, v^{(L)})$ and $(u^{(R)}, v^{(R)})$ correspond to the projection of (x, y, z) in the $\mathcal{I}$ space of the left and right cameras respectively, represent the zero disparity surface of the stereo system, called *horopter* [26]. Whenever a stereo system acquires images of an object that matches the horopter *size*, *shape*, and *position*, the two stereo images become identical. In this case, the search for homologous points becomes trivial, since they have the same coordinates in both stereo images. This property is extremely useful when the horopter coincides with the model of the feature of interest: in fact, the detection of disparities between the model and real scene under these circumstances is reduced to a simple pixel-wise comparison between the two acquired images. As mentioned above, for example, in the detection of obstacles that can obstruct the vehicle's path it may be useful to overlap the horopter with the planar surface representing the ground.

As shown in figure 4.22, the horopter has a spherical shape, the smaller the angle formed by the cameras' optical axes (*cameras' vergence*), the larger the radius [26]; when the cameras' vergence is small the horopter can be considered planar. Unfortunately, since only the horopter's *size* (radius) can be modified acting on cameras' vergence, the horopter cannot be overlapped with a *generic* surface, because this would also require changing its *position* and *shape*.

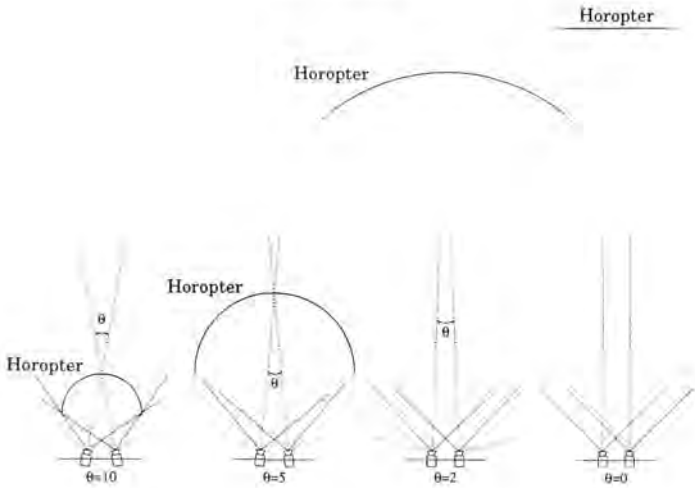


Fig. 4.22 Relationship between the angle formed by the optical axes of two stereo cameras and the horopter.

IPM can solve this problem, acting as an *electronic vergence*, therefore permitting the overlapping of the horopter with any surface. More precisely, since IPM can be used to recover the texture of a specific surface (the xy plane in the example of road markings detection), when applied to both stereo images (with different parameters reflecting the different acquisition setup of the two cameras) it provides two instances, namely two partially overlapping patches, of the given surface, consequently modifying the horopter's *shape*. These two patches, thanks to the knowledge of the vision system setup, can be brought to a correspondence, so that the horopter's *position* is also changed, and homologous points share the same coordinates in the two remapped images.

4.3.2.1 An application of stereo IPM: Obstacle Detection

As mentioned above, when Obstacle Detection means the mere *localization* of objects that can obstruct the vehicle's path without their complete *identification* or *recognition*, stereo IPM can be used in conjunction with a geometrical model of the road in front of the vehicle [57]. Assuming the

flat road hypothesis introduced in the previous section, IPM is performed using the same relations expressed by equations (4.13). This is of basic importance since in a system aimed at both Obstacle and Lane Detection the IPM transform may be carried out only once and its result can be shared by the two processes. The flat road model is checked through a pixel-wise difference between the two remapped images: in correspondence to a *generic obstacle* in front of the vehicle, namely anything rising up from the road surface, the difference image features sufficiently large clusters of non-zero pixels that have a particular shape.

In fact, the IPM transform maps straight lines perpendicular to the road surface into straight lines passing through the projection $C_{xy} \equiv (l, d, 0)$ of the camera onto the $z = 0$ plane (see figure 4.20). Using formulas (4.11), a vertical straight line is represented by the set of pixels

$$\left\{ P(u_i, v_i) \mid v_i = \bar{v} = \text{constant} \right\}; \quad (4.14)$$

thus, defining

$$\begin{cases} k_{\bar{v}} & \triangleq (\bar{\gamma} - \alpha) + \bar{v} \frac{2\alpha}{n-1} = \text{constant} \\ p(u) & \triangleq \tan \left[(\bar{\theta} - \alpha) + u \frac{2\alpha}{n-1} \right] \end{cases}, \quad (4.15)$$

it follows that

$$\begin{cases} x(u, \bar{v}) & = \frac{h}{p(u)} \cos k_{\bar{v}} + l \\ y(u, \bar{v}) & = \frac{h}{p(u)} \sin k_{\bar{v}} + d \\ z & = 0 \end{cases}, \quad (4.16)$$

which represents the straight line $x = (y - d) \cot(k_{\bar{v}}) + l$ passing through $C_{xy} \equiv (l, d, 0)$. Therefore the IPM transform maps the vertical edges of generic obstacles into straight lines intersecting in C_{xy} . Since in a stereo vision system the projections of the cameras on S do not coincide, due to the stereo cameras' different angles of view, an ideal homogeneous square obstacle produces two clusters of pixels with a triangular shape in the difference image, in correspondence to its vertical edges.

This behavior is shown in figure 4.23, which depicts the left and right views of an ideal white square obstacle on a gridded dark background, the two corresponding remapped images, and the thresholded difference also showing the overlapping between the two viewing areas.

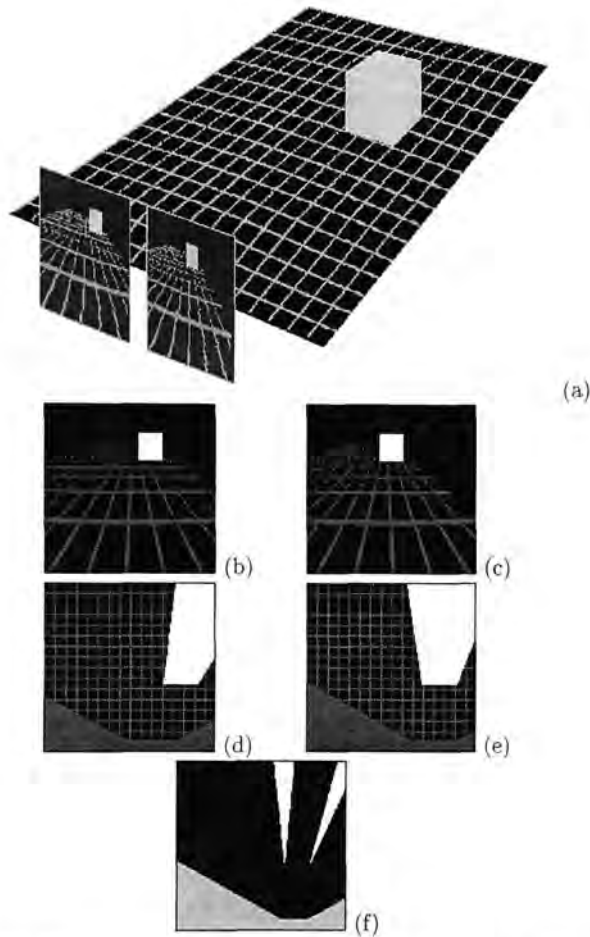


Fig. 4.23 The acquisition of an ideal homogeneous square obstacle: (a) 3D representation; (b) left image; (c) right image; (d) left remapped image; (e) right remapped image; (f) difference image in which the grey area represents the region of the road unseen by both cameras.

4.3.3 Functionalities

In the following section the Lane Detection and Obstacle Detection functionalities are discussed. Both of them are divided into a low-level phase that can be efficiently expressed with an SIMD computational paradigm and a serial high- and medium-level phase.

4.3.3.1 Lane Detection

Lane Detection is performed assuming that a road marking in the $z = 0$ plane of the $\mathcal{W}$ space (i.e. in the remapped image) is represented by a quasi-vertical bright line of constant width surrounded by a darker region (the road). Consequently, the pixels belonging to a road marking have a higher brightness value than their left and right neighbors at a given horizontal distance. Figure 4.24.a and 4.24.b show the original and remapped images together with a linear brightness profile of a horizontal line.

The first phase of road markings detection is therefore based on a line-wise determination of horizontal black-white-black transitions, while the following medium-level process is aimed at the extracting information and reconstructing road geometry.

Feature Extraction

The brightness value $b(x, y)$ of a generic pixel belonging to the remapped image is compared to its horizontal left and right neighbors at a distance m : $b(x, y - m)$ and $b(x, y + m)$, with $m \geq 1$.

A new image, whose values $r(x, y)$ encode the presence of a road marking, is then computed according to the following expression:

$$r(x, y) = \begin{cases} d_{+m}(x, y) + d_{-m}(x, y) & \text{if } (d_{+m}(x, y) > 0) \wedge (d_{-m}(x, y) > 0) \\ 0 & \text{otherwise} \end{cases} \quad (4.17)$$

where

$$\begin{cases} d_{+m}(x, y) = b(x, y) - b(x, y + m) \\ d_{-m}(x, y) = b(x, y) - b(x, y - m) \end{cases} \quad (4.18)$$

represent the horizontal brightness gradient. The choice of m depends on the road markings width, image acquisition process, and parameters used in the remapping phase. Considering $m = 2$, the filtered image is shown in figure 4.24.c.

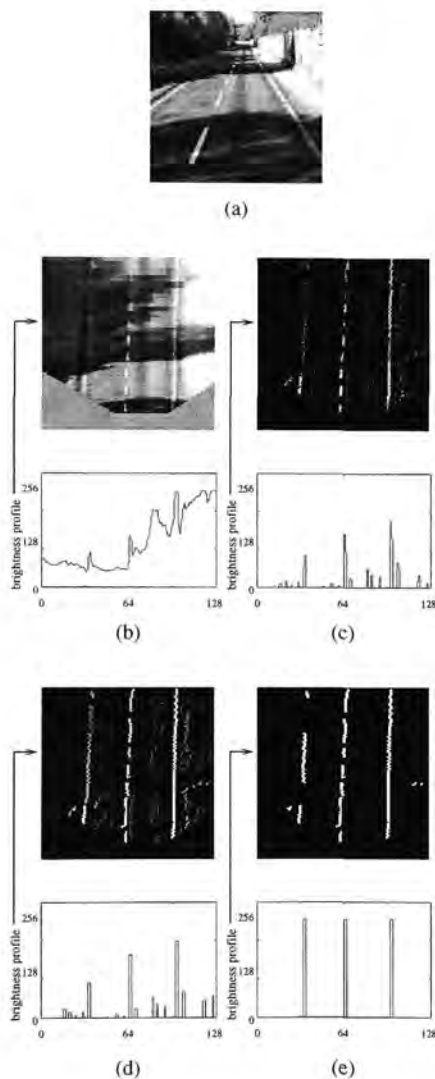


Fig. 4.24 The sequence of images produced by the low-level Lane Detection phase together with a linear profile of one of their lines: (a) original; (b) remapped; (c) filtered;

Due to different light conditions (e.g. in presence of shadows), pixels representing road markings may have different brightness, yet maintaining their superiority relationship with their horizontal neighbors. Consequently, since a simple threshold seldom gives a satisfactory binarization, the image is enhanced exploiting its vertical correlation; then an adaptive binarization is performed.

The enhancement of the filtered image is performed through a few iterations of a *geodesic morphological dilation* [135] with the following binary structuring element

$$\begin{bmatrix} & \bullet & \\ \bullet & \bullet & \bullet \\ & \bullet & \end{bmatrix}, \text{ where}$$

$$c(x, y) = \begin{cases} 1 & , \text{ if } r(x, y) \neq 0 \\ 0 & , \text{ otherwise} \end{cases} \quad (4.19)$$

is the *control image* [135]. The result of the geodesic dilation is the product between the control image and the maximum value computed amongst all the pixels belonging to the neighborhood described by the structuring element. The iterative application of the geodesic dilation corresponds to the widening of the neighborhood, except towards the directions in which the control values are 0. Figure 4.25 shows the results of 3 iterations on a portion of an image that represents a lane marking.

Moreover, since

$$r(x, y) \neq 0 \implies \begin{cases} r(x, y - m) = 0 \\ r(x, y + m) = 0 \end{cases}, \quad (4.20)$$

the filtered image's pixels at a distance m from a road marking assume a zero value; according to equation (4.19) their control value is 0 and as a result they form a barrier to the propagation of the maximum value. The enhanced image (after 8 iterations) is shown in figure 4.24.d.

Finally, the binarization is performed by means of an adaptive threshold:

$$t(x, y) = \begin{cases} 1 & , \text{ if } e(x, y) \geq \frac{m(x, y)}{k} \\ 0 & , \text{ otherwise} \end{cases}, \quad (4.21)$$

where $e(x, y)$ represents the enhanced image, $m(x, y)$ the maximum value computed in a given $c \times c$ neighborhood, and k is a constant. The result of the binarization of figure 4.24.d (considering $k = 2$ and $c = 7$) is presented in figure 4.24.e.

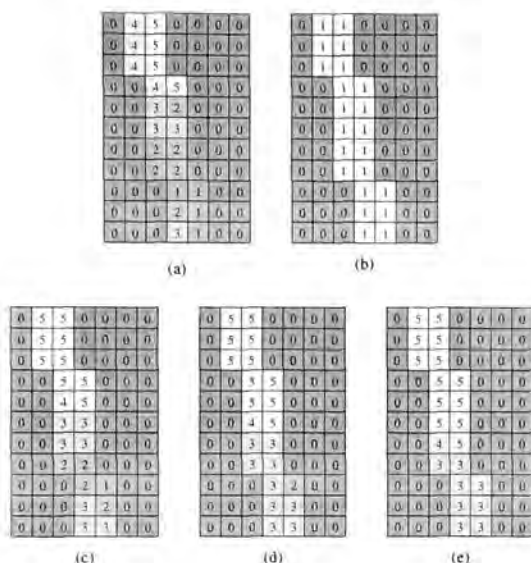


Fig. 4.25 The geodesic morphological dilation used for image enhancement: (a) input image, (b) control image, and (c-e) results of the first three iterations.

Road Geometry Reconstruction

The binary image is scanned row by row in order to build chains of 8-connected non-zero pixels. Following the choice of $m = 2$ at the previous step, every non-zero pixel can have at the most one adjacent non-zero pixel in the horizontal direction; as a result for each pair of horizontally adjacent pixels the one farthest to the right is chosen so that chains are built with a thickness of one pixel.

Each chain is approximated with a *polyline* made of one or more segments, by means of an iterative process. Firstly, the two extrema (A and B) of the polyline are determined averaging the position of the last few pixels at either end of the chain, in order to cope with possible ripples. The segment that joins these two extrema is considered. A first approximant segment ($\overline{AB}$), built using these two extrema, is considered and the horizontal distance between its mid point M and the chain is used to determine the quality of the approximation. In case it is larger than a threshold, two

segments $(\overline{AM'})$ and $(\overline{M'B})$ sharing an extremum are considered for the approximation of the chain, where M' is the intersection between the chain and the horizontal line that passes through M (see figure 4.26.a). The process is iterated on the approximant segments $\overline{AM'}$ and $\overline{M'B}$; at the end of the processing all chains will be approximated by polylines.

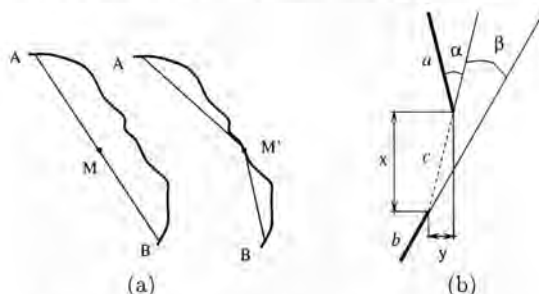


Fig. 4.26 (a) Approximation of chains with polylines and (b) parameters considered for polylines joining.

To get rid of possible occlusions or errors caused by either noise or ineffective low-level processing, two or more polylines are joined onto longer ones taking in account different parameters, such as: the horizontal distance x and the vertical distance y between the nearest extrema, the angles α and β formed by the final segments a and b of each polyline respectively and the segment c that joins the nearest extrema of the polylines (see figure 4.26.b). Since the parameters taken into account depend only on the ending segments of each pair of polylines, the joined polylines need to be evaluated also globally: those featuring a too high or too variable curvature and those that are shorter than a given threshold are discarded. When more solutions exist for joining the polylines, all of them are considered.

A reference line is used to select the polyline which most likely matches the road center line. Initially the vehicle is assumed to be in a specific position (center of the lane) on the road, which, at the same time, is thought of as being almost straight. In this situation the road center line in the remapped image is a straight vertical line that is expected to be found in a circumscribed area of the remapped image. Computed polylines are matched against this reference line using several parameters such as distance, parallelism, orientation, and length. The polyline that fits better

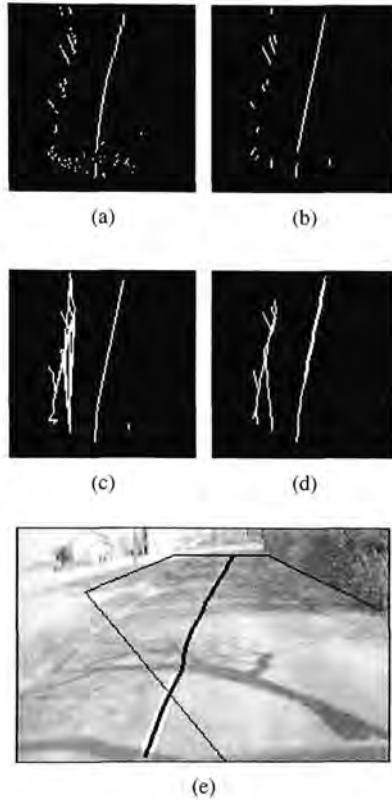


Fig. 4.27 The medium- and high-level steps of Lane Detection: (a) concatenation of pixels; (b) segmentation and construction of polylines; (c) joined polylines; (d) filtered polylines; (e) superimposition of the detected center line onto a brighter version of the original image for displaying purposes.

these parameters is selected and then used to determine a new reference, enabling the system to track the center line in image sequences and also to work with bent roads. A different initialization of the reference line is available which makes the system track the right lane marking instead of the road center line.

Since the model assumed for the external environment (flat road) al-

lows to determine the spatial relationship between image pixels and the 3D world [94], it is possible to derive from the previous result both the road geometry and vehicle position within the lane. Figure 4.27 shows the steps of Lane Detection.

Some results of Lane Detection in a number of different conditions are shown in figure 4.28.

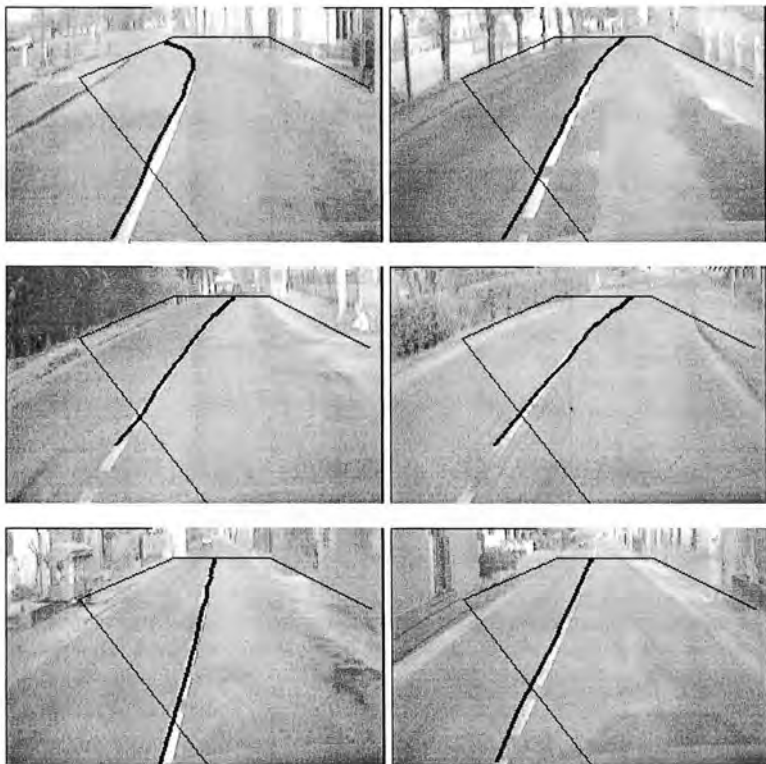


Fig. 4.28 A few results of Lane Detection on a rural road.

4.3.3.2 Obstacle Detection

As shown in section 4.3.2, the stereo IPM technique can produce a difference image in which ideal square obstacles are transformed into two triangles. The Obstacle Detection process is based on the localization of pairs of these triangles by means of a quantitative measurement of shape and position of the two corresponding triangles. Unfortunately due to the texture, irregular shape, and non-homogeneous brightness of real obstacles, the detection and localization of the triangles becomes difficult. Nevertheless, in the difference image some clusters of pixels with a quasi-triangular shape are anyway recognizable, even if they are not clearly disjointed. Moreover, in case two or more obstacles are present in the scene at the same time, more than two triangles appear in the difference image. A further problem is caused by partially visible obstacles which produce a single triangle.

The low-level portion of the process, detailed in figure 4.29, is consequently reduced to the computation of difference between the two remapped images, a threshold, and a morphological opening [70] aimed at removing small-sized details in the thresholded image.

Polar Histogram

A *polar histogram* is used for the detection of triangles: given a point F in the $z = 0$ plane of the W space (called *focus*), it is computed scanning the difference image and counting the number of overthreshold pixels for every straight line originating from the focus F . The polar histogram's values are then normalized using the polar histogram obtained by scanning an image where all pixels are set (*reference image*). Furthermore, a low-pass filter is applied in order to decrease the influence of noise (see figure 4.30).

When F lies on the prolongation of a triangle edge, the polar histogram presents a sharp discontinuity. The localization of a triangle is therefore performed exploiting the following property: the prolongations of the edges of a triangle intersect both projections $C_{xy}^{(L)}$ and $C_{xy}^{(R)}$ of the two cameras onto the road plane, as shown in figure 4.31.

Due to the short distance between $C_{xy}^{(L)}$ and $C_{xy}^{(R)}$, instead of computing two different polar histograms (having focus on $C_{xy}^{(L)}$ and $C_{xy}^{(R)}$) a single one is considered. The polar histogram's focus is placed in the middle of $\overline{C_{xy}^{(L)} C_{xy}^{(R)}}$; in this case the polar histogram presents an appreciable peak corresponding to each triangle. Since the presence of an obstacle produces two disjointed triangles (corresponding to its edges) in the difference image,

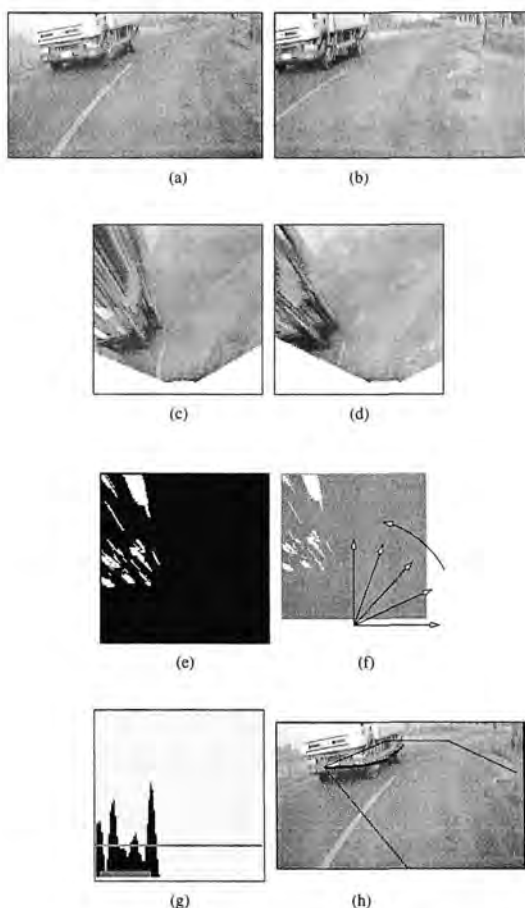


Fig. 4.29 Obstacle Detection: (a) left and (b) right stereo images, (c) and (d) the remapped images, (e) the difference image, (f) the angles of view overlapped with the difference image, (g) the polar histogram, and (h) the result of Obstacle Detection using a black marking superimposed on the acquired left image; the thin black line highlights the road region visible from both cameras.

Obstacle Detection is limited to the search for pairs of adjacent peaks. The position of a peak in fact determines the angle of view under which the

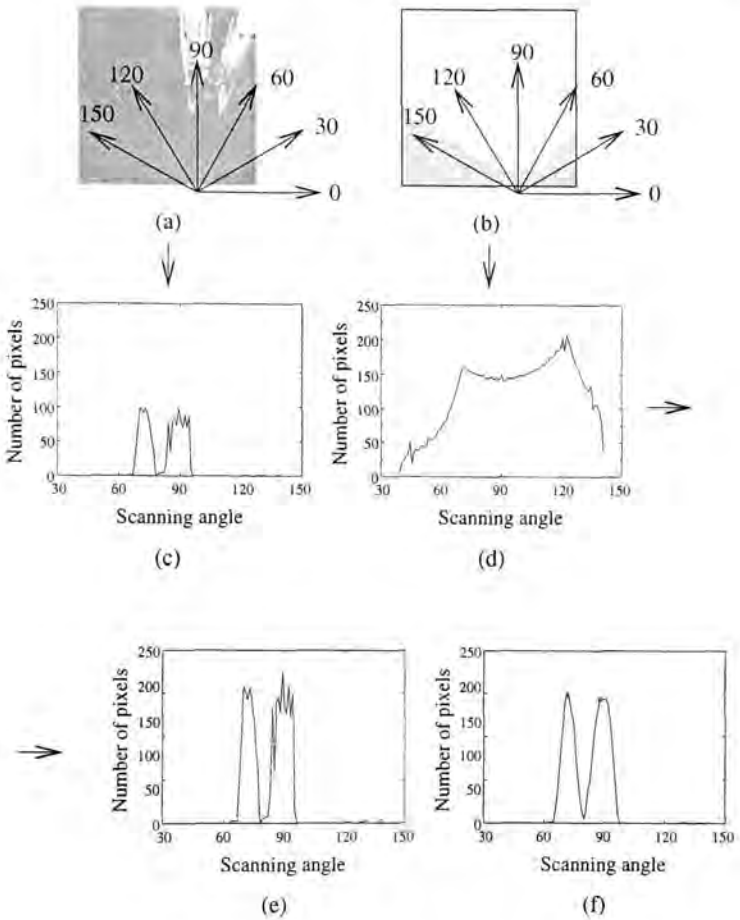


Fig. 4.30 Steps involved in the computation of the polar histogram: (a) original difference image; (b) reference image; (c) polar histogram computed from the difference image; (d) normalizing factor obtained scanning the reference image; (e) normalized histogram; (f) filtered polar histogram.

obstacle edge is seen (figure 4.32).

Peaks may have different characteristics, such as amplitude, sharpness,

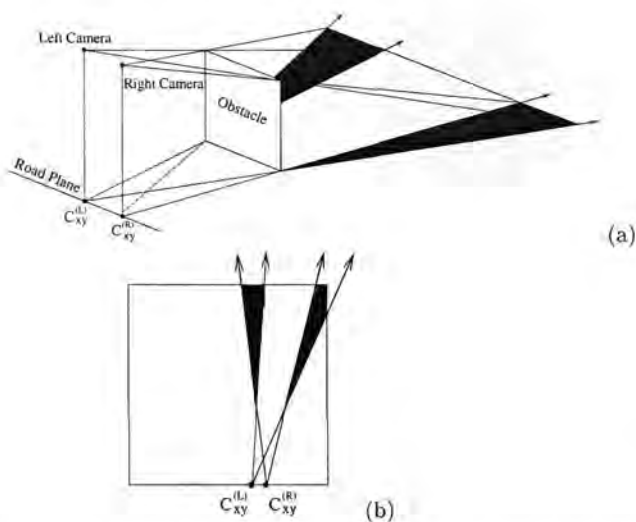


Fig. 4.31 Relationships between obstacle and cameras position: (a) perspective view and (b) road plane view.



Fig. 4.32 Correspondence between triangles and directions pointed out by peaks detected in the polar histogram.

or width. This depends on the obstacle distance, angle of view, and difference of brightness and texture between the background and the obstacle itself (see figure 4.33).

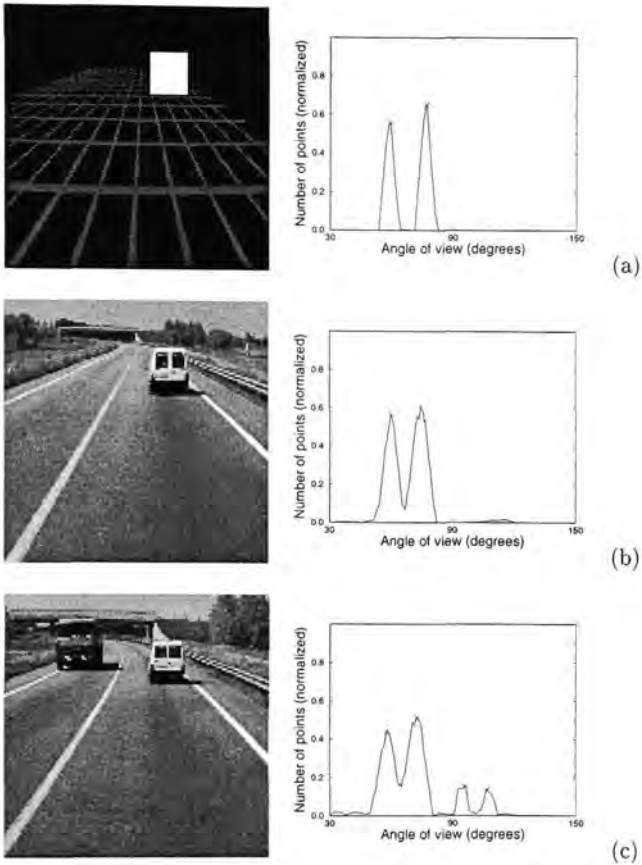


Fig. 4.33 Polar histograms: (a) ideal situation, (b) real situation with one obstacle and (c) real situation with two obstacles.

Peaks Joining

Two or more peaks can be joined according to different criteria, such as similar amplitude, closeness, or sharpness. The analysis of a large number of different situations made possible the determination of a parameter embedding all of the above quantities. According to the notations of fig-

ure 4.34, R is defined as the ratio between areas A_1 and A_2 . If R is greater than a threshold, two adjacent peaks are considered as generated by the same obstacle, and then joined; otherwise, when the two peaks are far apart or the valley is too deep they are left alone (not joined). Figure 4.35 shows some examples of peak joining.

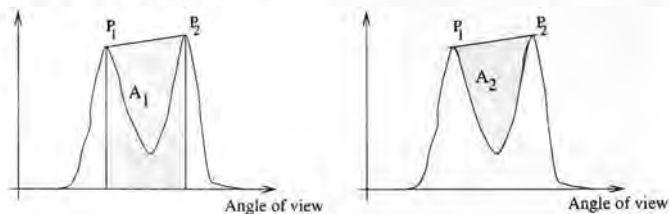


Fig. 4.34 If the ratio between areas A_1 and A_2 is greater than a threshold, the two peaks are joined.

Obviously, a partially visible obstacle produces a single peak that cannot be joined to any other. The amplitude and width of peaks, as well as the interval between joined peaks, are used to determine the angle of view under which the whole obstacle is seen.

Estimation of Obstacle Distance

The difference image can also be used to estimate the obstacle distance. For each peak of the polar histogram a *radial histogram* is computed scanning a specific sector of the difference image. The width α_i of the sector is determined as the width of the polar histogram peak in correspondence to 80% of the peak maximum amplitude h_i . The number of overthreshold pixels is computed and the result is normalized. The radial histogram is analyzed to detect the corners of triangles, which represent the contact points between obstacles and road plane, therefore allowing the determination of the obstacle distance through a simple threshold (see figure 4.36).

The result is displayed with black markings superimposed on a brighter version of the left image; they encode both the obstacles' distance and width. Figure 4.37 shows the results obtained in a number of different situations.

Figure 4.38 presents the results of both Obstacle Detection and Lane Detection.

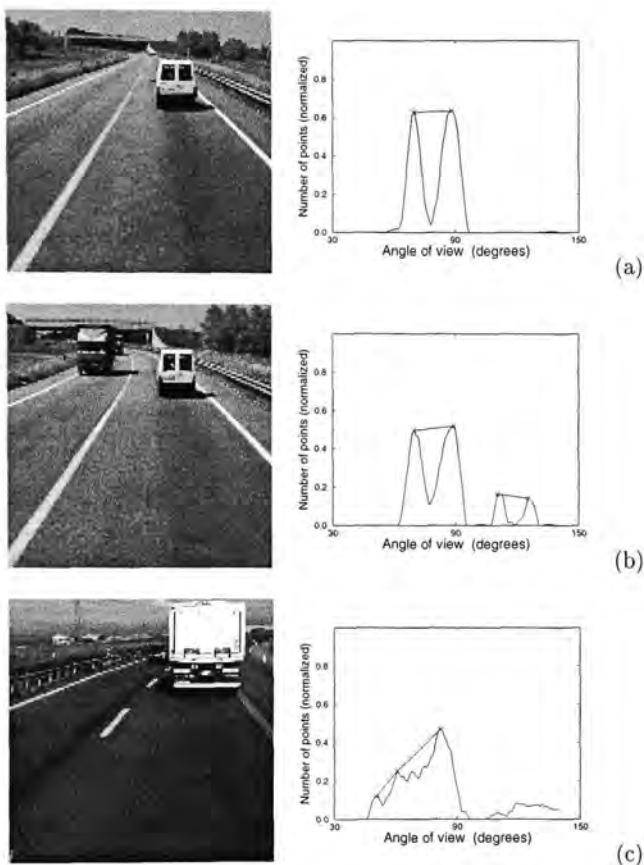


Fig. 4.35 Some examples of peaks join: (a) one obstacle, (b) two obstacles and (c) a large obstacle.

4.3.4 An extension of Inverse Perspective Mapping to handle non-flat roads

A large number of implementations of autonomous driving systems developed worldwide [6, 32, 118, 158], including the previously described GOLD

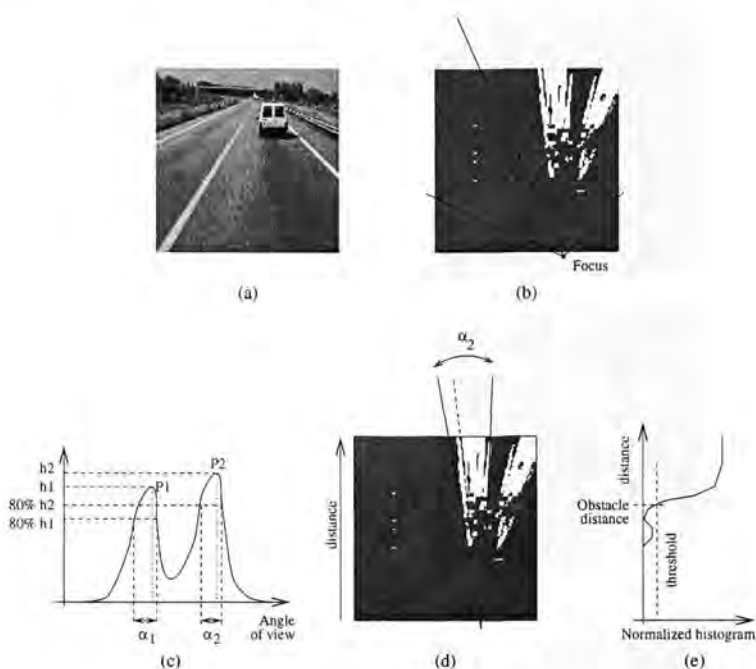


Fig. 4.36 Steps involved during the computation of radial histogram for peak P_2 : (a) original image; (b) binary difference image; (c) polar histogram; (d) sector used for the computation of the radial histogram; (e) radial histogram.

system, rely on the strong assumption of moving on a flat road or –more realistically– on a road with a smoothly varying slope. This assumption is of fundamental importance since in the great majority of these systems the first stage of the processing is based on the Inverse Perspective Mapping. Obviously, the formulas that define this mapping heavily rely on the knowledge of the road surface's geometry, which is not always completely known. It can mainly vary according to the following two different reasons:

- vehicle movements (pitch and roll), which change the reference system of the acquisition device with respect to the road;
- changes in the slope of the road.

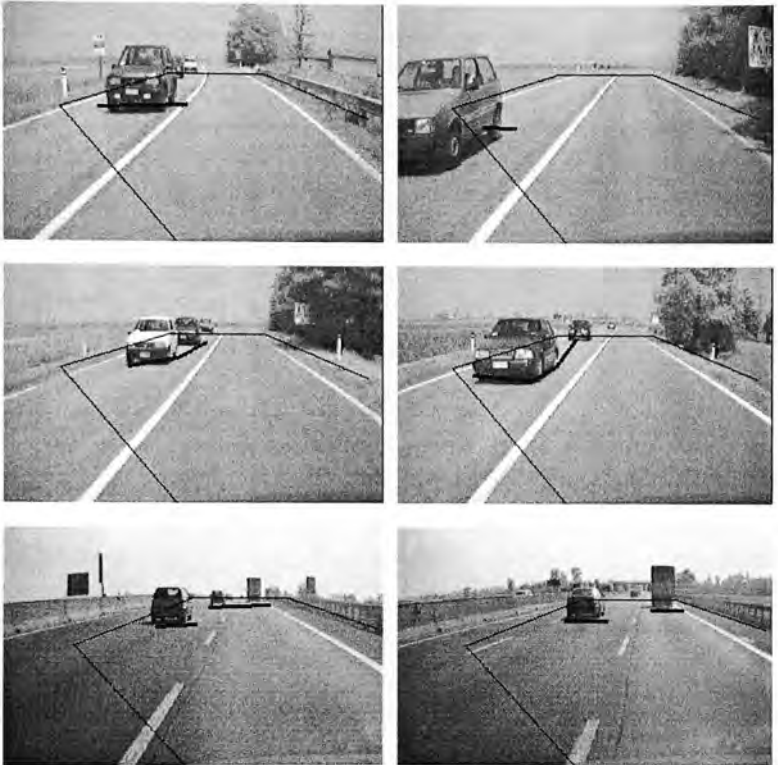


Fig. 4.37 Obstacle Detection: the result is shown by a white marking superimposed on the image captured by the left camera; a black thin line limits the portion of the road seen by both cameras.

While minor vehicle movements (generally at a low frequency, about 1 Hz) and small changes in the road slope can be neglected for relatively short fields of view (as it will be shown in section 4.3.5), appreciable deviations from the flat road assumption may lead to unacceptable distortion in the resulting image.

Although during the *MilleMiglia in Automatico* tour (see chapter 7) the behavior of the system was satisfactory throughout the whole trip, in some

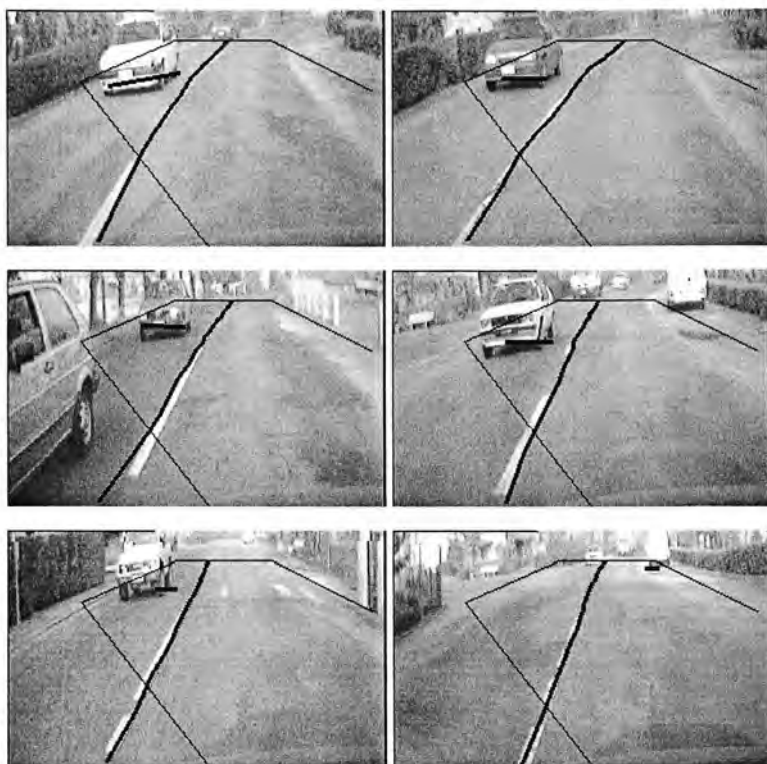


Fig. 4.38 A few results of both Obstacle and Lane Detection.

specific situations –mainly in the Appennine regions, where many subsequent changes in the road slope are present– the assumption of a flat road was not met and some calibration errors occurred. These errors led to wrong measurements of the vehicle position, generating lateral oscillations of the vehicle within the lane during uphill and downhill driving. A quantitative measurement of these lateral movements was not carried out but it was estimated to be less than one meter. This effect can be tolerated on highways where the lanes have a large width, but can be critical on roads with narrow lanes (typical Italian extra-urban roads).

Therefore an extension of the IPM technique is required to also handle roads whose slopes can vary quite abruptly. This extension (Extended Inverse Perspective Mapping, EIPM) relies on the processing of stereo images, whose analysis is used to update the road slope ahead of the vehicle, and is based on the following steps:

- (1) both stereo images are processed (remapped) with the IPM transform, using the current set of parameters;
- (2) from the two new images some features are extracted, which belong to the road surface (in the current implementation these features are the road markings);
- (3) if these features have the same coordinates in both images, then the remapping parameters used by the IPM are correct and reflect the current acquisition setup and road slope;
- (4) in case the features have different coordinates, a recalibration is required and the correct road slope must be computed. The new slope is measured from the coordinate differences and the IPM parameters are updated.

In the following sections the whole technique will be described referring to synthetic images. Figures 4.39.a and 4.39.b show a pair of synthetic stereo images of a flat road, while figures 4.39.c and 4.39.d show their remapped versions. Thanks to a correct calibration and to the absence of any obstacle, the two remapped images are identical. On the contrary, when the calibration does not reflect the actual road slope (i.e. the flat road assumption is not met, as in figures 4.39.e and 4.39.f) the two remapped images differ, as shown in figures 4.39.g and 4.39.h.

4.3.4.1 *Extraction of features belonging to the road plane*

In order to refine the calibration, corresponding features belonging to the road plane must be extracted from both remapped images and their relative positions compared. The Obstacle Detection step is used to cut the portion of the images representing the obstacles. This process guarantees that the remaining features belong to the road plane.

The feature extraction process is based on the localization of lane markings, since they belong to the road plane. This approach does not represent an overhead since this process is already carried out for the road following task (Lane Detection).

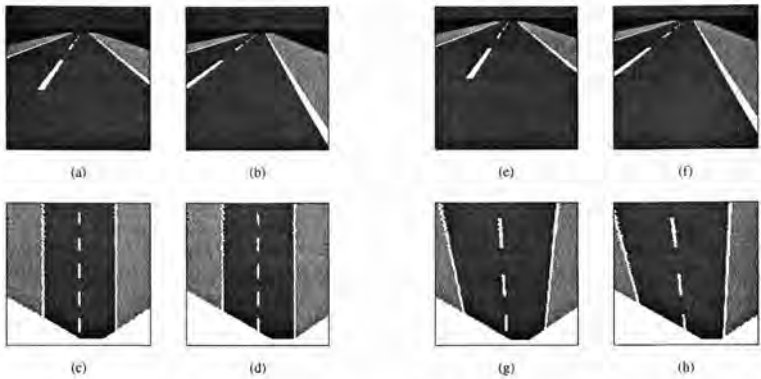


Fig. 4.39 (a) and (b) synthetic stereo images of a flat road, (c) and (d) their remapped versions; (e) and (f) synthetic stereo images of a non flat road, (g) and (h) their remapped versions.

4.3.4.2 Determination of road slope

The determination of road slope is based on the following assumptions:

- the two stereo cameras are installed at the same height;
- the roll angle of both cameras is null;
- the results of the previous step represent homologous features belonging to the road plane.

Figure 4.40 represents the vision system's setup and shows the projections on the $z = 0$ plane, with respect to the two cameras, of a generic point P of coordinates (x, y, z) , with $z \neq 0$. P_R and P_L represent the points to which the point P is remapped by the IPM transform.

Called d the distance between L_0 and R_0 (the projections of the two cameras onto the $z = 0$ plane), δ the distance between P_R and P_L , and h the height of the cameras with respect to the $z = 0$ plane, thanks to the geometrical similarity between triangles $L_0 \triangle P_L$ and $P \triangle P_0 P_L$ and between triangles $L_P R$ and $P_L \triangle P P_R$, it follows

$$z = h \frac{\delta}{d + \delta} . \quad (4.22)$$

Since h and d are known, z can be determined by measuring δ . Therefore, analyzing the positions of pairs of homologous points in the remapped im-

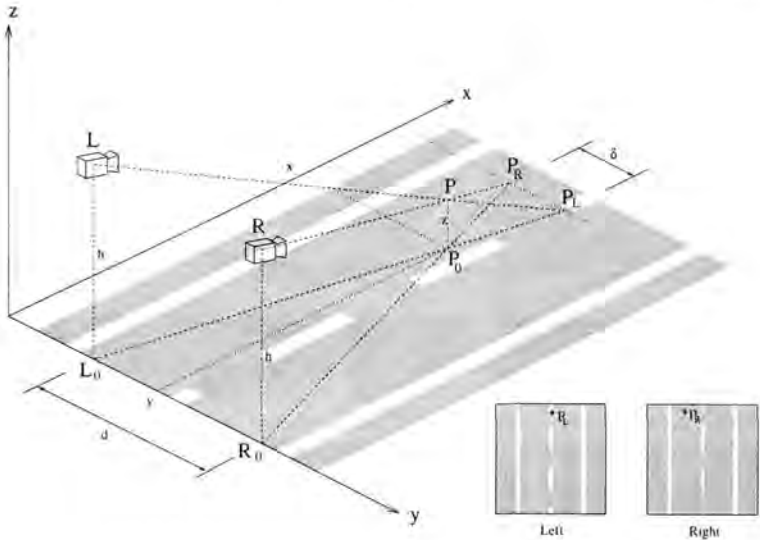


Fig. 4.40 The determination of the height of a generic point P.

ages it is possible to determine the coordinates of the point they represent in the 3D space. In particular, in case the two points have the same coordinates ($\delta = 0$), it follows that $z = 0$ (this happens for each point belonging to the lane markings when the road is flat).

In this way it is possible to compute the "height" of each point representing a road marking, i.e. its z coordinate. The series of these values yield the road slope ahead of the vehicle.

Figure 4.41 shows the following: the remapped images obtained from a pair of synthetic stereo images representing a 2% road slant assuming a flat road, the extraction of the center line from both of them, their different position and a measurement of the road slope.

4.3.4.3 System recalibration

Although it is possible to compute the road slope ahead of the vehicle as a sequence of values representing the height of all lane marking points, the slope is approximated with a linear function. This approximation is computed as an average of these values.

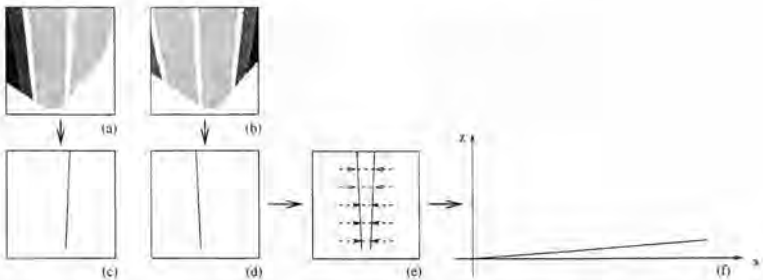


Fig. 4.41 Determination of the road slope for a 2% slant: (a) and (b) remapped images; (c) and (d) extraction of the center line; (e) superimposition of the previous two images, showing the different position of the two center lines; (f) graphical output of the computed linear road slope.

This average value will be used for the recalibration of the system as a correction to the cameras' tilt angle. The effectiveness of this approximation is now being tested on real images.

4.3.4.4 Performance analysis

In figure 4.42 the results of the application of the EIPM technique to a number of synthetic images are presented. The left remapped images before and after the recalibration step are shown with different values of the road slope (ranging from +5% to -5%). Thanks to the EIPM, the final remapped images represent the texture of the road without distortions. This proves that the system is able to extract the correct slope of the road.

This extension possesses an advantage that does not rely on complex computations, and its computational overhead is limited to the match between a small number of coordinates and the correction of the calibration parameters. In fact, the extraction of features belonging to the road plane is already performed by the Lane Detection functionality, while the EIPM is only based on a further analysis of these results.

4.3.5 Discussion

The GOLD system was initially tested with images acquired by the stereo vision system which was installed on-board the experimental vehicle MOB-LAB. These tests allowed to analyze the system's performance and

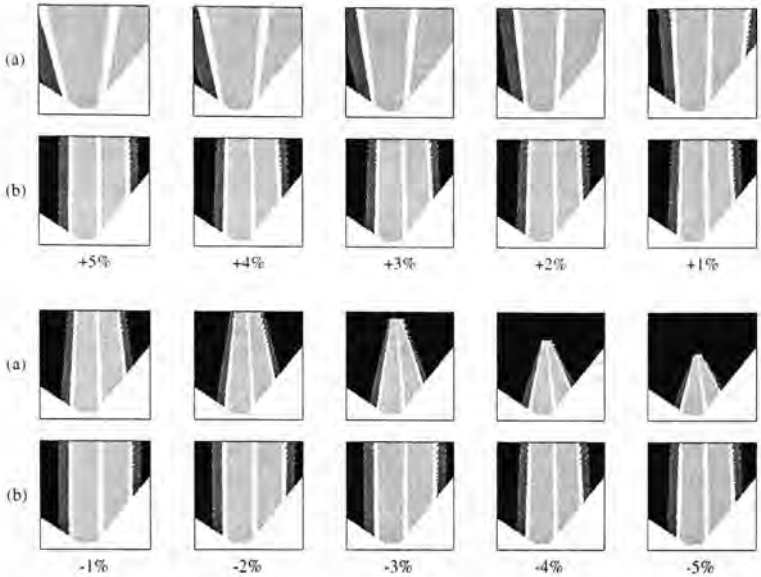


Fig. 4.42 (a) images remapped with the IPM using the set of calibration parameters obtained assuming a flat road. (b) images remapped with the EIPM using the set of corrected parameters. This figure shows the behavior for different values of the road slope ranging from +5% to -5%.

to detect its critical points.

Since Obstacle Detection uses stereo vision, the quality of the results is also tightly coupled with the vision system's calibration. Nevertheless, since the final target of Obstacle Detection means determining free space in front of the vehicle and not is the complete 3D reconstruction of the world, camera calibration becomes less critical. For this reason, even if the vehicle's movements modify some of the calibration parameters (camera height h and inclination $\bar{\theta}$ with respect to the road plane), a dynamic recalibration of the system is not required. For comparison purposes, ranging values for the cameras' height ($h \pm 10$ cm) and inclination ($\bar{\theta} \pm 1^\circ$) higher than the ones estimated in [90] have been considered. Figures 4.43 and 4.44 demonstrate the results of Obstacle Detection emulating the changes of the cameras parameters caused by vehicle movements. Thanks to the robustness of the approach based on the polar histogram, the obstacle is always

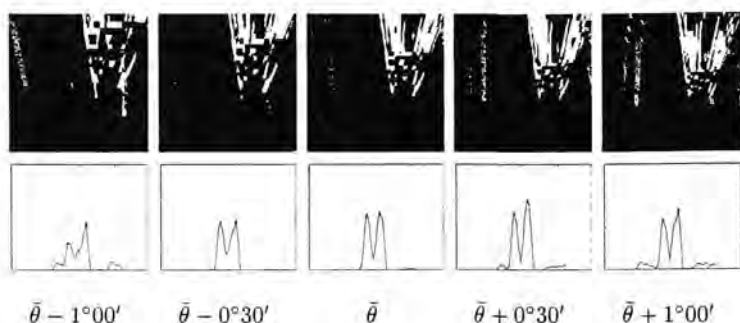


Fig. 4.43 Obstacle Detection changing the inclination parameter: difference image and polar histogram for different values of the inclination parameter.

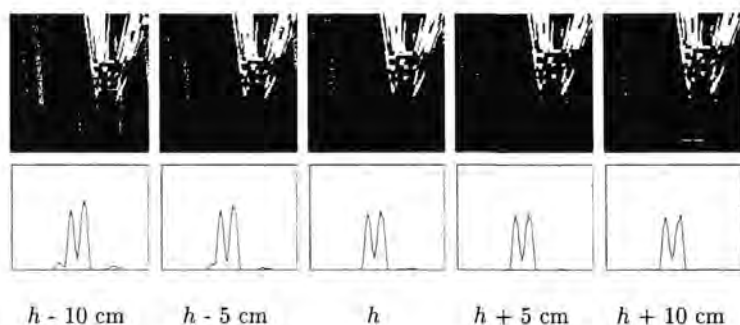


Fig. 4.44 Obstacle Detection changing the height parameter: difference image and polar histogram for different values of the height parameter.

detected even if the difference images are noisy.

The major critical problems of Obstacle Detection are found when:

- the obstacle is too far from the cameras (this generally happens in the $45 \div 50$ m range), in this manner the polar histogram presents only small and isolated peaks that can be hardly joined (figures 4.45.a, 4.45.b, and 4.45.c); however, this problem has never been detected when the obstacles' distance is in the $5 \div 45$ m range;
- the guard-rail is close to the obstacle and for this reason a single large obstacle is detected (figure 4.45.d);

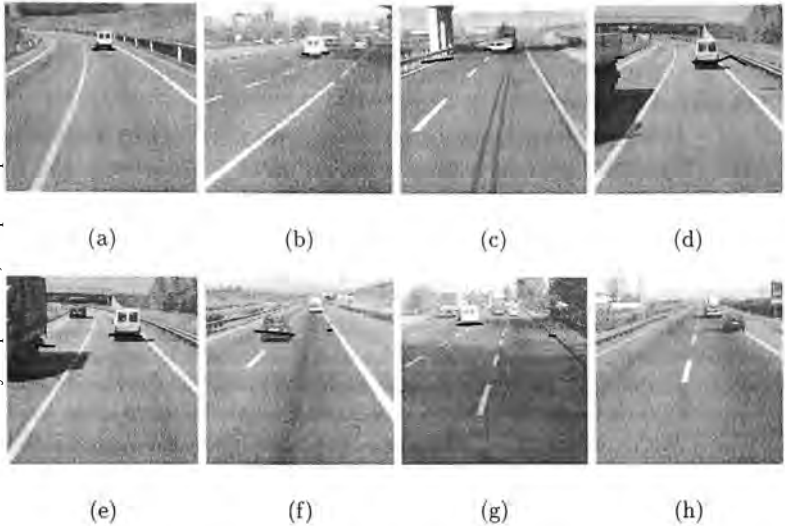


Fig. 4.45 Situations in which Obstacle Detection is critical.

- an obstacle is partially visible, and consequently only one of its edges can be detected (figures 4.45.d and 4.45.e);
- some noisy peaks in the polar histogram are not filtered out, and they are therefore considered as small obstacles (figures 4.45.f and 4.45.g);
- the detection of far obstacles sometimes fails when their brightness is similar to the brightness of the road (figure 4.45.h).

The confidence achieved in the detection of obstacles is obviously dependent on their size, distance, and shape, and specific parameters are used to tune the system's sensitivity.

Obstacle height

The obstacle height determines the amplitude of peaks in the polar histogram. The bandwidth of the low pass filter applied to the polar histogram is the parameter used as threshold to discard small peaks that could be caused by either noise or short obstacles. The smaller the bandwidth, the lower the influence of noise (caused by incorrect

camera calibration or vehicle movements), but higher is the minimum height of detectable obstacles (in figure 4.45.c and 4.45.d the guard-rail is detected even if it is not as tall as vehicles).

Obstacle width

In the polar histogram two or more peaks generated by the same obstacle are joined together when the ratio $\frac{A_1}{A_2}$ is greater than a threshold. In the worst case (when the peaks are separated by deep valleys), the smaller the threshold, the larger is the width of the wider correctly detectable object, but higher is the probability that peaks not generated by the same obstacle are joined (see figure 4.45.d).

Obstacle distance

The farther the obstacle, the smaller the portion of triangles detectable in the difference image, and therefore the lower is the amplitude of peaks in the polar histogram. Nevertheless, for sufficiently high obstacles (e.g. vehicles at about 50 m away from the cameras) the main problem is not the detection of peaks, but their joining, as shown in figure 4.45.a, 4.45.b, and 4.45.c.

Obstacle shape

The algorithm was designed to detect obstacles with quasi-vertical edges. Objects with non-vertical edges (e.g. pyramidal objects) generate twisted triangles that are hardly detected through the analysis of the polar histogram.

The presence of the ARGO vehicle at the *Dipartimento di Ingegneria dell'Informazione, Università di Parma*, enabled further tests of the GOLD system to be carried out. Since only a portion of the incoming image is analyzed, a different setup of the stereo cameras has been chosen on the ARGO vehicle compared to MOB-LAB, in order to both increase the resolution of the remapped image and solve the problems caused by framing a portion of a bright sky (see figure 4.46). Also the inter-camera spacing is a key parameter: the greater the distance between cameras, the stronger the disparities in the remapped images due to the presence of an obstacle. Unfortunately, large separations lead to a higher sensitivity towards vehicle movements, in particular rolling.

During the tests, the system demonstrated to be robust and reliable: vehicles were always detected and only in a few cases (i.e. on paved or—more generally—rough roads) vehicle movements became so strong that

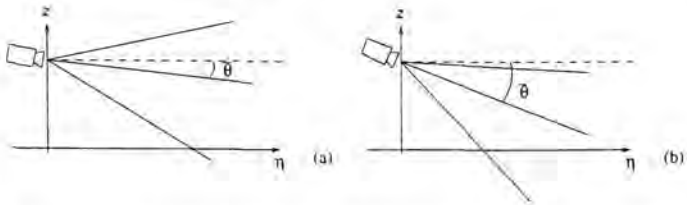


Fig. 4.46 (a) Setup of the vision system on the MOB-LAB vehicle; (b) setup of the vision system on the ARGON vehicle.

the processing of noisy remapped images led to the erroneous detection of false small-sized obstacles. Nevertheless, since the vehicle movements have a high frequency, these small-sized obstacles only appear in a limited number of consecutive frames, and can be easily removed thanks to a temporal averaging filter.

On the other hand, thanks to the remapping process, lane markings were located even in the presence of shadows or other artifacts on the road surface. Although it is hard to devise a method to evaluate the percentage of successful Lane Detection, some unofficial tests have shown that the system detects the correct position of the lane in about 95% of the situations considered; however, this result does not take into account an exhaustive set of road conditions.

4.3.5.1 A cooperative approach

This section presents an extension of the GOLD system able to exploit also temporal correlations and perform a deeper data-fusion between the two functionalities of Lane and Obstacle Detection.

Once obstacles have been detected and localized, the knowledge of their position can be of great help to the Lane Detection module, as obstacles generally obstruct the visibility of road markings: the regions occluded by obstacles will not be considered while searching for road markings, consequently decreasing noisy features that may disturb the retrieval of road geometry.

Similarly, assuming the knowledge of the road geometry (namely the position of road markings within the image), and the precise position of obstacles, a pair of stereo images can be analyzed to validate the assumption of planarity regarding road surface. Furthermore, the stereo pair can be

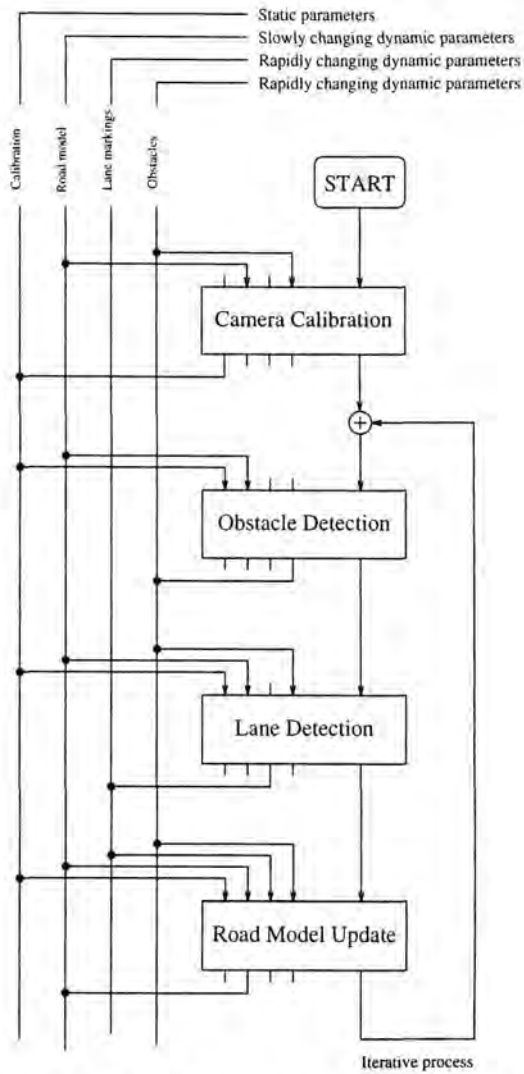


Fig. 4.47 Block diagram of the cooperative approach: the left part depicts the data stream, while the right part shows the control flow of the algorithm.

used to calibrate the stereo system and adapt it to new road surfaces, such as hills, bridges, or non-planar highway ramps; the new model will replace the old one and will be used by both Obstacle and Lane Detection modules.

These considerations led to the development of an integrated cooperative approach in which the Obstacle and Lane Detection modules are triggered off alternatively; each module exploits the results produced by the previous module and feeds the following one with its own results, therefore keeping an updated data structure describing the three *dynamic* parameters:

- obstacles position,
- road geometry,
- and road surface.

This iterative process is initialized by hand when no obstacles are visible and the road in front of the vehicle is flat; the initialization phase is used to determine the acquisition system's calibration, i.e. the *static* parameters.

More precisely, this integrated approach, depicted in figure 4.47, is described as follows.

System Calibration

The system needs to be initialized under known conditions. Since the calibration of the acquisition system consists in determining the angles formed by the two cameras' optical axes with vertical and horizontal directions, the geometry of the road surface in front of the vehicle must be known. Therefore the calibration is triggered by hand when the portion of the road visible by both cameras is flat and no obstacle is present. This phase takes 30 ÷ 40 seconds, since an exhaustive search in the parameters' space is carried out [57]: each parameter (i.e. angles determining the cameras' orientations) scans a given interval and for each parameters configuration the two remapped stereo images are computed according to a flat road model. The set of parameters for which the two remapped stereo images present the minimum pixel-wise difference is chosen to be the correct cameras' calibration, which remains fixed for the whole run.

Obstacle Detection

Using the given road model and knowledge of the acquisition system parameters, obstacles are detected as discussed in section 4.3.3.2.

Lane Detection

The left image is processed as discussed in section 4.3.3.1 and the road geometry extracted.

Road Model Update

The results of the previous step indicates where lane markings are located or, in case of occlusions caused by obstacles, where they are expected to be.

Assuming that the obstacles' position is known within the cameras field of view, the portion of lane markings that has been generated to overcome occlusions can be discarded. Therefore the remaining part identifies features that belong to the road surface. These features are particularly important because they represent horizontal brightness transitions (lane markings) in the left remapped image and the search for correspondent patterns in the right remapped image (on the same epipolar line [58]) allows to determine the *height* of the road markings with respect to the original $z = 0$ plane used for calibration.

Since lane markings belong to the road surface, this step is used to update the internal model of the road surface. The computation of the height of a rather large number of points that represent road markings, followed by a temporal average, permits filtering and compensating noise caused by sudden vehicle movements. On the other hand, the use of a focus of attention limited only to points that represent road markings, simplifies and speeds-up the processing.

Chapter 5

Hardware Support for Real-Time Image Processing

This chapter contains the presentation of some architectural solutions for real-time image processing in automotive applications.

Firstly the PAPRICA system –a massively parallel custom architecture specifically developed for real-time image processing– is described together with a specific hardware support for the removal of the perspective effect; then its main bottlenecks are highlighted and a new architecture, PAPRICA-3, is presented. After the description of the hardware system, the implementation of the main steps for Obstacle Detection is discussed.

Finally, the evolution towards a general-purpose commercial platform –Intel Pentium with MMX technology– is motivated and the implementations of Lane Detection and Obstacle Detection algorithms on the different architectures are presented, compared, and analyzed.

The chapter ends with some remarks about architectural trends in the field of image processing and, in particular, for automotive applications.

5.1 The PAPRICA Architecture

This section presents the PAPRICA (PARallel PRocessor for Image Checking and Analysis) architecture jointly developed by the *Dipartimento di Ingegneria dell'Informazione, Università di Parma* and the *Dipartimento di Elettronica, Politecnico di Torino*.

5.1.1 Architectural issues

In the field of image processing for automotive applications, the response time of the system is a major critical point, since it affects directly the maximum speed allowed for the vehicle, and therefore the choice of the computing architecture is a key design issue. At the beginning of the project the performance of commercial systems was too low to allow real-time image processing. Therefore a lot of special-purpose solutions were designed and built, most of which were based on massively parallel architectures composed of a large number of relatively slow-clocked Processing Elements (PEs). This choice was originally motivated by the consideration made regarding power consumption: the power consumption of dynamic systems, in fact, can be considered proportional to CfV^2 , where C represents the capacitance of the circuit, f is the clock frequency, and V is the voltage swing. Power can be saved in three different ways [59], minimizing C , f , and V respectively:

- (1) using a greater level of VLSI integration, thus reducing the capacitance C ;
- (2) trading computer speed (with a lower clock frequency f) for lower power consumption (already implemented on many portable PCs);
- (3) reducing the supply voltage V_{DD} .

Different technological solutions have been exploited to reduce the IC supply voltage, but unfortunately there is a speed penalty to pay for this reduction. For a CMOS gate [136], the device delay T_d (following a first order approximation) is proportional to $\frac{V_{DD}}{(V_{DD} - V_T)^2}$, which demonstrates that the reduction of V_{DD} determines a quasi-linear increment (up to the device threshold value V_T) of the circuit delay T_d . On the other hand, the reduction of V_{DD} determines a quadratic reduction of the power consumption. Therefore, for power saving reasons, it is desirable to operate at the lowest possible speed, but, in order to maintain the overall system performance, compensation for these increased delays is required.

The use of a low power supply voltage is investigated in [31,38], where different architectural solutions are considered in order to overcome the undesired side effects caused by the reduction of V_{DD} . The reduction of power consumption, while maintaining computational power, can be reached by the use of low cost SIMD computer architectures, formed by a large number of extremely simple and relatively slow-clocked PEs. These systems, using

slower device speeds, provide an effective mechanism to trade power consumption for the silicon area, while maintaining the computational power unchanged. The 4 major drawbacks of this approach are:

- (1) a solution based on hardware replication increases the silicon area, and it is therefore not suited for extremely area-constrained designs;
- (2) parallelism must be accompanied by extra-routing, requiring extra-power; this issue must be carefully considered and optimized;
- (3) the use of parallel computer architectures involves the redesign of the algorithms with a different computational model;
- (4) since the number of processing units must be high, if the system has size constraints the PEs must be extremely simple, performing only simple basic operations.

On the other hand, besides sufficient computational power, the features that allow the integration and efficient use of a computer system on board a moving vehicle are: (i) low production cost, (ii) low operative cost, and (iii) small physical size.

5.1.2 Hardware system description

The PAPRICA system [23] is a low-cost special-purpose massively parallel architecture composed of 256 PEs working in SIMD fashion. It was developed following the above requirements and its design dates back to 1990. The Processing Array (PA) is composed of an array of 4×4 full custom ICs ($1.5 \mu\text{m}$ CMOS, 45 mm^2 , $\simeq 35,000$ transistors), each of them containing a sub-array of 4×4 PEs. The PA is a 16×16 square matrix of 1-bit PEs each one with a total 8-neighbors connectivity [60, 106, 123]; each PE has an internal memory composed of 64 bits; and data flow between the Image Memory and the PA through a 16-bit data bus. The PAPRICA coprocessor is integrated on a single VME board (6U) connected to a SPARC-based host workstation. It is comprised of 5 major functional parts: *Program Memory*, storing up to 256k instructions; *Image Memory* (IM, up to 8 Mbyte of static RAM), whose 16-bit memory fetch time is 150 ns; *Processor Array*, whose PE cycle time is 250 ns; *Camera Interface*, able to acquire stereo images and display temporary results on an external monitor at video rate (25 frames/s or 50 fields/s); and a *Control Unit*, managing the activity of the whole system.

Generally in image processing applications, the number of PEs is far

smaller than the number of image pixels, therefore requiring the *virtualization* of the PA. This is done utilizing an external Image Memory [106, 123, 132]: the PA is loaded from the external memory with a sub-window of the data set; then the computation is performed until a special instruction is reached; finally, the results are stored back again into the external memory. These steps are iterated until all the sub-windows have been processed. Then the first sub-window is reloaded again into the PA and the computation is resumed.

A number of features are integrated into the PAPRICA system for the efficient implementation of the driving assistance functions. These are: the morphological processing paradigm, which is fundamental for low-level processing; the hardware support for pyramidal vision tasks; the direct integration of an image acquisition device; and the hardware support for a fast image remapping.

5.1.2.1 Morphological processing

The PAPRICA instruction set is comprised of two kinds of operations: graphical and logical. Graphical operators derive from mathematical morphology [70, 135], which is a bit-map approach to image processing based on set theory. For each pixel, the input to the graphical operator comes from one bit-plane of the pixel itself and from the *same* bit-plane of its eight surrounding neighbors: the output value depends on the particular morphological operator. Operators are provided to perform: *inversion*, *shift* in the four main directions, global, vertical, and horizontal *dilation* and *erosion*, and *border extraction*. The result of the graphical operator is then either stored in a destination bit-plane or used as the first operand of the following logical operation.

Sources of the logical operator are the result of the graphical operator (the central pixel itself, if no operation was performed) and the value of a second bit-plane of the same pixel. The most common operators (NOT, AND, OR, EXOR) are provided; in addition, the instruction set comprises a sum operator allowing multi-bit computations.

5.1.2.2 Support for pyramidal vision

Pyramidal architectures have shown several advantages in the field of image processing [27, 129, 144]. One major drawback of such architectures is that they are not easily scalable beyond a certain size, because interconnec-

tions amongst processors have an intrinsic 3D structure which cannot be easily mapped onto a 2D silicon surface. Thanks to the specific processor virtualization mechanism, PAPRICA system allows the implementation of multi-resolution algorithms with no dedicated hardware interconnections.

5.1.2.3 The camera interface

A major critical point of the first PAPRICA prototype was the time required to acquire and display images since no direct camera interface was present. This bottleneck has been removed by integrating the image acquisition device directly on board. This allows both grabbing and direct storing of pairs of grey-tone (8 bit/pixel) stereo images into PAPRICA image memory and to display the contents of the image memory on a monitor at video rate in two formats: 512×512 pixel @ 25 Hz, corresponding to full frames, and 512×256 pixel @ 50 Hz, corresponding to single fields (even or odd).

5.1.2.4 Data remapping support

A communication is defined to be *global* when the destination coordinates within the image are different for every source pixel. With a 2D mesh interconnection topology it is impossible to perform global communications directly (and efficiently). The PAPRICA system includes a specific hardware mechanism that allows the global movement of the image pixels without any constraints on the final destination.

An *address* image is loaded into the Image Memory and treated as a look-up-table; each pixel contains a pointer to a memory location where the source value for that pixel resides. A dedicated serial device, implemented in the same FPGA integrating PAPRICA controller, scans the address image and creates the new remapped image, whose pixel values are determined by an indirect addressing.

The remapping process takes three 50 ns clock cycles per pixel giving a total of about 3 ms to remap a 128×128 image. This device is fundamentally important for the transformation described in section 4.3.1 and for its applications described in section 4.3.3.

5.2 Critical Analysis of the PAPRICA Architecture

The availability of the first PAPRICA prototype allowed the carrying out of performance measures and a critical analysis of the whole system. The most important bottlenecks were identified and their possible solutions considered.

In the following section, the main problems observed are discussed in detail. The first one, specific to massively parallel architectures, refers to processor virtualization and to the physical allocation of the memory; the second one is related to the I/O limitation; finally, the external architecture of the PEs is also discussed.

5.2.1 Memory organization and processor virtualization

The limitation existing for the amount of internal memory (64 bits) of each PE is the main bottleneck regarding performance for two basic reasons.

- It is not possible to store internally a large amount of intermediate results, as it is often needed by computation intensive problems, therefore requiring I/O operations to store the intermediate data externally.
- It is not possible to implement the well-known virtualization mechanism adopted by the other parallel architectures like the Connection Machine CM-2 [145], where the computation is serialized within each processor (whose memory contains data belonging to several pixels).

Whereas the first problem is a real limitation, the latter can be solved using the virtualization mechanism described in section 5.1.2, which, on the other hand, offers some specific advantages. The PAPRICA virtualization mechanism is less efficient than the one implemented on the CM-2 in terms of speed since it requires a lot of accesses to an external memory, but it offers the possibility to simulate a pyramidal structure [144] on the 2D mesh without any other hardware modification, and to implement a kind of *focus of attention* [107, 159].

Pyramidal emulation: the PAPRICA image memory can be rearranged run-time using special instructions, which set the image height, width and depth. Moreover, it is possible to take advantage of

the fact that for each parallel computation $2Q^2$ (where Q is the size of the PA) sequential accesses to the IM are required to fetch and store back the data in a useful way. In fact, the data to be transferred into the PA may be not logically adjacent to each other in the IM, allowing to undersample the image or to increase its resolution. This behavior is controlled by a set of registers which can be altered run-time. The possibility to reduce and increase the image dimensions allows a very efficient use of PAPRICA architecture as a pyramid. Moreover, some simple software algorithms also allow to simulate any kind of pyramidal interconnections between the different pyramid layers.

Focus of attention: An additional 64-bit register and simple logics allow each sub-window elaboration to be performed conditionally to one or more bits of the register. The implementation of this hardware extension does not involve major changes in the existing controller, since it is programmed on a XILINX module. The efficient implementation of this sort of *multiple focus of attention* is possible thanks to the PAPRICA PE virtualization mechanism. The triggering idea is based on the consideration that sometimes, after the loading of an image sub-window into the PA, the elaboration may be useless, and therefore the operation of storing back the results into the image memory may not be required. The choice whether to perform the elaboration or not depends on the characteristics of the complete set of data loaded into the PA internal registers.

5.2.2 I/O problems

The data transfer between the image memory and the processing array in the present version takes place on a common 16-bit data bus. The addressing mechanism is such that a bus cycle selects a single PE to read or write a block of 16 bits belonging to the same pixel. This solution was chosen in the early stage of the design to simplify the architecture of the processing array chip, but it has proven to be somewhat inefficient. In fact, due to the windowing mechanism, in order to process a block of instructions it is always necessary to previously load the PEs with data from an image sub-window and to store them back after processing. Both data transfers have to be carried out on blocks of 16 image planes, while in many cases (almost all the practical ones) the number of necessary input and output

planes is not a multiple of 16. This means that a significant amount of time is wasted transferring unnecessary data back and forth from the IM.

The efficiency of the system can then be improved changing the addressing mechanism to allow an access by image plane instead of the present one: a bus cycle will then transfer 16 bits belonging to the same image plane from/to 16 different PEs. It has to be noted however that it is difficult to implement this transfer method with the present architecture: in fact, the sub-windows of the image are not a multiple of 16 bits if graphic operators have to be processed. This means that, to maintain consistency, the controller has to shift and merge data from different locations during loading and storing operations. The added delay would reduce consistently what is gained by optimizing data organization.

5.2.3 Instruction set

The PAPRICA computational paradigm is based on the concept of the *matching operator*, derived from the *hit-miss transform* described in [135]. This is a rather general approach which includes the other morphological operators as special cases. PAPRICA can perform matching operations using a fixed set of structuring elements, which form the *array instruction set*. This instruction set can implement any structuring element using simple compositions following the mathematical morphology algebra.

PAPRICA has been conceived as a CISC processor. Subsequently, the evaluation of the possible applications of PAPRICA has led to the consideration that the instruction set should evolve towards a RISC implementation, allowing the execution of a minimum set of simple operations, such as translations of binary layers and logic operations between layers. Since the complexity of the instruction set determines the PE size, the RISC choice allows the integration of a higher number of PEs on a single chip.

Furthermore, the possibility to enlarge the neighborhood involved in morphological operations has been investigated. With a relatively small PA, the execution of a large number of consecutive morphological operators requires a lot of I/O transfers (since there is a high overlapping between the sub-windows), decreasing the computational speed. With the increment of the PA size, long sequences of graphic operators become feasible. On the other hand, the hardware complexity required to implement a larger neighborhood would increase the performance by a negligible amount. The RISC approach, implementing complex instructions as sequences of simple

ones, only increases the length of the program and not its complexity.

5.2.4 Architectural evolution

Since the first prototype implementation, the PAPRICA system has undergone a series of revision stages and a thorough investigation of a number of architectural modifications to overcome the previously described limitations. Basically the feasibility studies have outlined three main possible solutions which will be briefly described in this section.

- The first solution is to maintain the same architecture of the original prototype and increase the size of the array, taking advantage of the technological evolution and of the possible optimizations in the chip layout. If the size becomes of the same order of magnitude of the images to be processed, the loading/unloading overhead due to the overlapping of windows required by the use of morphological operators would be reduced.
- The second one is to increase the addressing space of each elementary processor by the use of external memory elements. This choice logically simplifies the processor virtualization mechanism but limits the array size and presents additional implementation problems due to morphological operators.
- The third type is to change the physical interconnection topology of the processing elements from a two-dimensional mesh to a linear array moving from a *window*-based to a *scanline*-based processor virtualization.

The problems encountered when trying to increase the system's performance, as exposed in the previous sections, are mainly due to the processor virtualization mechanism: it is always necessary to divide the image into sub-windows and, in order to execute morphological operators, it is necessary to know the contents of the pixels at the border of the sub-windows.

This problem is simplified if one of the image dimensions can be entirely loaded into the array, because in this case it is only necessary to scan the image in the other dimension. This is the basic reason which triggered the idea of investigating a new architecture based on a linear array of PEs, big enough to process a complete image line in one iteration [132]. The linear architecture also matches better the applications in real-time vision systems, where processing speed and data latency are the most critical

factors. Such systems have several characteristics that have to be considered to optimize the low-level processor:

- data arrives serially from a sensor which generally scans the image by rows, so data becomes available one row at a time;
- image dimensions are fixed and dictated by the resolution of the sensor;
- the results of low-level processing must be further elaborated by other stages managing higher level tasks.

In this environment, a one-dimensional array is the best solution to minimize data latency, because a row can be processed as soon as it is acquired from the sensor. Furthermore, the image width in such systems normally ranges from 128 to 1024 pixels, making it perfectly possible to match the width of the array to that of the image.

The proposed linear topology is implemented in the PAPRICA-3 architecture discussed in the next section.

5.3 The PAPRICA-3 Architecture

The PAPRICA-3 architecture, jointly developed by the research units of the *Politecnico di Torino* and *Università di Parma*, is a real-time hardware accelerator for low-level image processing applications.

The PAPRICA-3 system, designed to be interfaced with a conventional microprocessor, is able to capture a serial input data stream coming from an external camera (or pair of stereo cameras), process it in parallel on a line-by-line basis, and store the resulting images either onto a dedicated dual-port memory, also accessible from the conventional processor, or onto a monitor.

The heart of PAPRICA-3 consists of a SIMD linear array of PEs, each one devoted to a single line pixel. Each PE can perform *logical* or *morphological* operations on an input pixel. Logical operations compute a one-bit result depending on values stored on different image planes of the same pixel, while morphological operations evaluate a one-bit result based on the values stored in the neighboring pixels on a single image plane.

The execution unit, which is common to the whole processor array, controls program flow and can perform tests, branches, conditional execution of program segments, loops, and synchronization statements. To speed up

program execution, a fairly complicated pipeline scheme has been adopted to exploit as much instruction level parallelism as possible. The program is completely stored in an internal program memory (512 instructions wide) writable from the external processor, so there is no need for internal cache memory.

A Status Register File can be used to compute and store *global parameters* of the processed images, such as statistics or degrees of matching between an input and reference images. It is accessible to the external processor and is aimed at increasing the execution speed of a broad class of neural algorithms.

PAPRICA-3 is a modular system. Each chip is composed of 32 PEs, a generic I/O video interface and does not include any internal image memory. Instead, the chip has a fast interface towards an external image memory and a Status Register File, in order to achieve two goals: connecting several chips together, it is possible to create a system with a varying number of PEs; an external but fast image memory will not decrease the performance too much compared with an internal memory, but the amount of memory can be much higher, to cope with memory demanding applications.

A programming environment has been developed to ease the prototyping of applications on PAPRICA-3. Application programs are written in C++; a code generator first converts C++ statements into assembly code; subsequently, optimization is performed on the generated assembly code. The optimizer performs both deterministic optimizations following both standard, well known techniques as well as stochastic ones to improve the performance of the pipeline structure and take advantage of the intrinsic instructions parallelism.

A detailed description of PAPRICA-3's programming environment can be found in appendix B.

5.3.1 Hardware system description

As shown in figure 5.1, the heart of PAPRICA-3 is a linear array of Q identical 1-bit PEs connected to an image memory via a bidirectional Q -bit wide bus. The image memory is organized in addressable words whose length matches that of the processor array; each word contains data relative to one binary pixel plane (also called *layer*) of one line of an image (Q -bit wide), and a single cycle is needed to load an entire line of data into the PE's internal registers. When executing a program, the correspondence

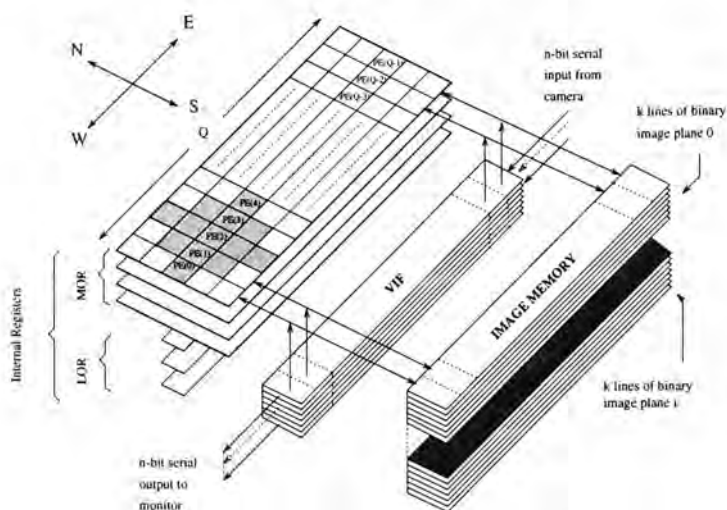


Fig. 5.1 General architecture of the Processor Array.

between the line number and the pixel plane of a given image and the absolute line address is computed in hardware by means of data structures, named *Image Descriptors*, stored in the Control Unit. An Image Descriptor is a memory pointer built up of three parts: a base address, a line counter and a page counter. Line and page counters can be reset or incremented by a specified amount under program control. Two counters, called M and N, are not directly related to image memory addressing, but can be used to modify program flow: instructions are provided to preset, increment and test them.

Data is transferred into the internal registers of each PE, then processed and explicitly stored back into memory according to a RISC-like processing paradigm.

Each PE processes one pixel of each line and is composed of a Register File and a 1-bit Execution Unit. The core of the instruction set is based on morphological operators: for each processor, the result of an operation depends on the value of pixels in a given neighborhood (reduced 5×5 , as sketched by the grey squares in figure 5.1). Data from E, EE, W and

WW directions (where EE and WW denote pixels two bits apart in E and W directions) may be obtained through the direct connection with neighboring PEs whilst all other directions correspond to data from previous lines (N, NE, NW, NN) or from subsequent lines (S, SE, SW, SS). Operations on larger neighborhoods can be decomposed [3] using chains of basic operations.

To obtain the outlined neighborhood, a number of internal registers (16, at present), called *Morphological Registers* (MOR), are used. They have a structure which is more complex than that of a simple memory cell, and are actually composed of five 1-bit cells with a S→N shift register connection. When a load operation from memory is performed, all data is shifted northwards by one position and the south-most position is taken by the new line from memory. In this way, data from a 5×5 neighborhood is available inside the array for each PE, at the expense of a two lines latency. A second set of registers (48, at present), called *Logical Registers* (LOR), are only 1-bit wide. On LORs and on the central bit of MORs it is possible to perform a different portion of the instruction set, which comprises logical and algebraic operations.

An important characteristic of the system is the integration of a serial-to-parallel I/O device, called *Video Interface* (VIF), which can be connected to a conventional input imaging device (camera, linear array) and/or to a monitor. The interface is composed of an 8-bit *Q*-stage shift register which serially and asynchronously loads/stores a new line of the input/output image during the processing of the previous/next line. Two instructions enable the bidirectional transfer between the PE's internal registers and the VIF, also ensuring proper synchronization with the camera and monitor.

Image processing algorithms consist of sequences of low-level steps, such as filters, convolutions, etc., to be performed line by line over the whole image. This means that the same block of instructions has to be repeated many times, and instruction fetching from an external memory is an important source of overhead. Therefore, each block of instructions is pre-loaded into an internal memory, named *Writable Control Store* (WCS, 512 words of 32 bits each), and instructions are fetched from there. In this way it is possible to obtain the performance of a fast cache with a hit ratio close to 1, at a fraction of the cost and complexity.

Two inter-processor communication mechanisms are also available to exchange information amongst PEs, which are not directly connected. The

first one is called the *Status Evaluation Network* (SEN), shown in figure 5.2, to which each processor sends the contents of one of its registers (selectable under program control) and which provides a *Status Word* divided into two subfields. The first one is composed of two global flags, named SET and RESET, which are true when the contents of the specified register are all 1 or all 0, respectively. The second subfield is the COUNT field which is set equal to the number of PEs in which the content of the specified register is 1. This global information may be accumulated and stored into the Status Register File and used to conditionally modify the program flow. As an alternative, it may be reloaded into the PE's Register File for further processing.

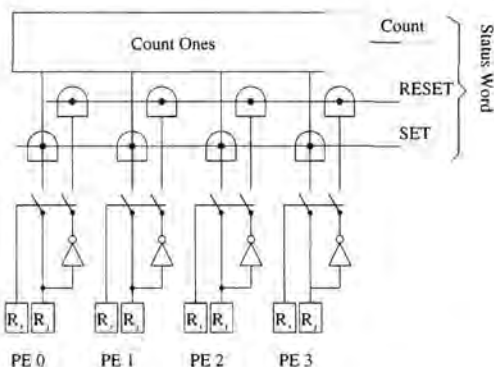


Fig. 5.2 The Status Evaluation Network.

The second communication mechanism is called the *Inter-processor Communication Network* (ICN), shown in figure 5.3, which allows global and multiple communication amongst clusters of PEs. The topology of the communication network may be varied during program execution. In fact, each PE drives a switch that enables or disables the connection between pairs of adjacent processors. Therefore, the PEs may be dynamically grouped into clusters, each PE can broadcast a register value to the whole cluster within a single instruction. This feature can be very useful in algorithms involving *seed-propagation* techniques, and in the emulation of pyramidal (hierarchical) processing.

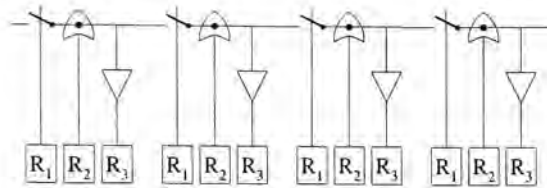


Fig. 5.3 The Interprocessor Communication Network.

5.3.2 Obstacle Detection on PAPRICA-3

This section describes how the Obstacle Detection algorithm presented in section 4.3.3.2 has been redesigned to exploit the line-wise processing paradigm of PAPRICA-3.

Since the geometrical transform used to remove the perspective effect requires the execution of complex data transfers between non-adjacent pixels, its implementation on PAPRICA-3 would suffer from low efficiency. Therefore only one image (the right one) is processed using a different geometrical transform: instead of removing the perspective effect, the right image is *warped* onto the left one. Assuming all the acquisition parameters known and a flat road in front of the vehicle, this warping produces a new image (*warped image*) that represents the scene as framed from the position of the other camera. If the flat road hypothesis is not met, the left and warped images differ. Since the following processing steps are simplified when the image to be processed is remapped in the road domain, the perspective effect is then removed from a *binarized* version of the difference image (1 bit deep). The analysis of this result leads to the determination of free space in front of the vehicle.

The advantage of this approach (summarized in figure 5.4.b) are the following:

- the warping of the right image can be approximately performed by a set of line shifts which match one of the PAPRICA-3's basic features;
- the removal of the perspective effect is only performed on one binary image (difference image) instead of two 8 bit deep grey level images (left and right images).

Figure 5.5 shows the sequence of images generated by this algorithm.

Downloaded from www.worldscientific.com
by 2601:282:700:6b90:5054:63cf:a8f2:87b6 on 12/22/24. Re-use and distribution is strictly not permitted, except for Open Access articles.

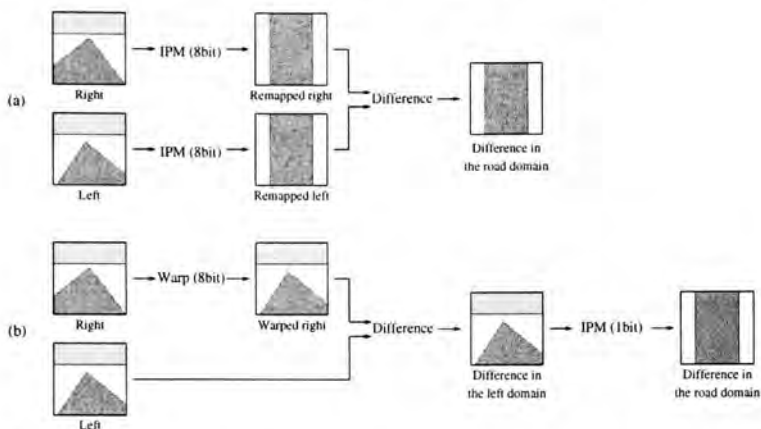


Fig. 5.4 The difference between (a) the original algorithm and (b) its PAPRICA-3 version.

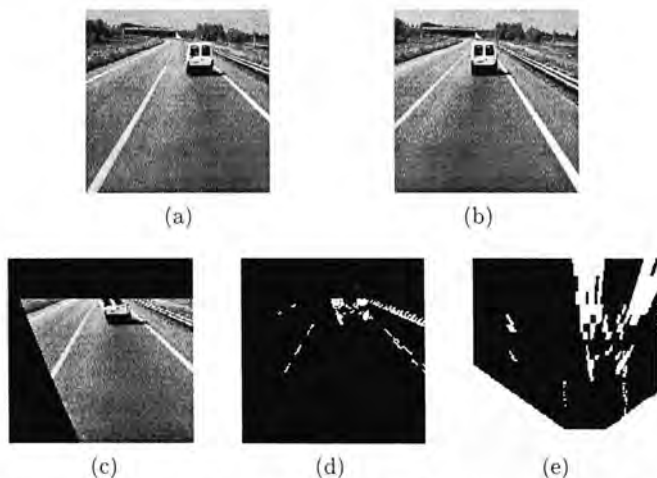


Fig. 5.5 Low-level Obstacle Detection on PAPRICA-3: (a) and (b) left and right acquired images, (c) right image warped onto the left one, (d) thresholded difference image, (e) difference image remapped onto the road plane.

5.3.2.1 Image warping

This section describes the warping procedure, namely the geometrical transform that, starting from the right image produces a new image representing the road region in front of the vehicle as if it were observed from the left camera.

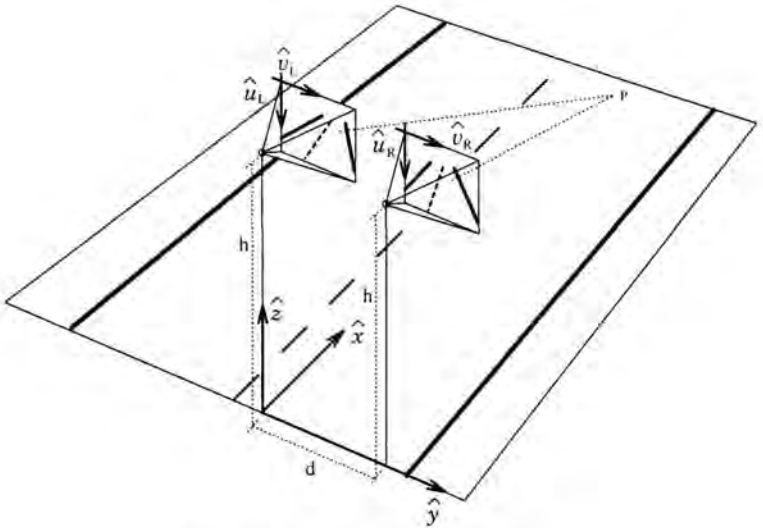


Fig. 5.6 The geometry of the stereo vision system.

The application of this transform requires knowledge of the following parameters (see figure 5.6):

- (1) the height h and distance d of the two cameras;
- (2) the viewing directions: for each camera the optical axis $\hat{o}$ is determined by the following angles: $\hat{\eta}$, the angle formed by the projection (defined by versor $\hat{\eta}$) of the optical axis $\hat{o}$ on the plane $z = 0$ and the x axis; and $\hat{\theta}$, the angle formed by the optical axis $\hat{o}$ and versor $\hat{\eta}$;
- (3) the cameras angular aperture 2α ;
- (4) the cameras resolution $n \times n$.

Considering $\hat{\theta}_L = \hat{\theta}_R = \hat{\theta}$, the relationship between the coordinates

(u_R, v_R) in the right image and the coordinates (u_L, v_L) in the left image is given by:

$$\begin{cases} u_R \approx u_L \\ v_R \approx v_L + \frac{n-1}{2\alpha} \left\{ \bar{\gamma}_L - \bar{\gamma}_R - \frac{d}{h} \tan \left[(\bar{\theta} - \alpha) + u_L \frac{2\alpha}{n-1} \right] \right\} = \\ = v_L + f(u_L) \end{cases} \quad (5.1)$$

The procedure defined by equation (5.1) produces a left view of the road scene starting from the right view. From the analysis of equation (5.1) it is visible that the warped image can be obtained by simply translating each line of the right image and that the amount of the translation depends on the row number.

Assuming a flat road and no obstacles in front of the vehicle, the left image and the warped one represent the *same* texture. Any difference between these two images (see figure 5.5.d) represents a deviation from the assumption of flat road, and therefore identifies an obstacle, i.e. anything rising up from the road surface. The following high-level Obstacle Detection process is therefore based on the analysis of the difference image, as described in section 4.3.3.2.

Parallel Implementation on PAPRICA-3

Since the warping procedure consists of an east line shift operation, it can be efficiently performed by PAPRICA-3 which is based on a linear array of PEs whose length matches the image width. Each PE obtains data thanks to a direct connection to its WEST and WEST WEST neighbors.

Thanks to the knowledge of the acquisition system setup, obtained by means of a calibration phase, the shift values for each image line are known at compile time and have been hardcoded into the PAPRICA-3 program, therefore leading to an execution time of about 0.2 ms for a 128×128 pixel 8-bit image.

5.3.2.2 Inverse Perspective Mapping on PAPRICA-3

After determining the disparities between the warped and left image, the resulting image is binarized and remapped in the road domain. The remapping process is accomplished by applying the Inverse Perspective Mapping transform discussed in section 4.3.1. The implementation of this transform requires the movement of pixels to/from arbitrary positions within the same

image line; moreover, the value of some pixels may be duplicated and sent to more than one new position, while the values of other pixels may be discarded. Fortunately, these movements, directly derived from the setup of the acquisition system, are completely known at compile-time and can be therefore hardcoded into the program. This fact eases the development of a parallel version of this transform, as discussed in appendix C.

In the PAPRICA-3 implementation, a single cycle of simultaneous communications requires the execution of about 30 Assembly instructions. Since PAPRICA-3's internal structure is based on a fairly complex pipeline, a specific ordering of the assembly instructions (using an *instruction scheduling* technique [121]) has been considered, using both traditional techniques [2] and the stochastic code optimizer discussed in appendix B. In the worst situation, depending on the specific set of compatible communications, the PAPRICA-3 system takes about 200 ns for the execution of a single set of binary communications.

In the remapping of a 128×128 pixel image, as shown in figure 5.7, a maximum of 19 cycles per line is required. Every single image line is therefore remapped in less than $\simeq 4 \mu\text{s}$, thus the total time required to remap a 128×128 pixel binary image is about 0,5 ms.

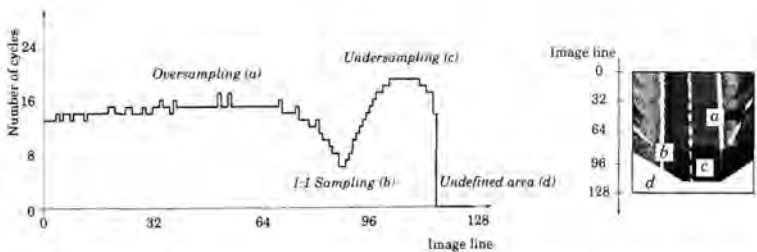


Fig. 5.7 The number of communication cycles required to remap a 128-pixel line as a function of the image lines' position in the remapped image. In the first phase (a) the original image lines are heavily oversampled, and the number of communication cycles remain almost constant; it is reduced to a minimum when (b) an almost 1:1 mapping is performed; and it grows up again in the undersampling phase (c); finally it is reduced to zero in the lowest undefined region (d). For clarity an 8 bit deep remapped image is shown.

5.4 The MMX Technology

In this section a description is given of MMX (Multi Media eXtensions or Matrix Math eXtensions) features [75] and programming techniques [74].

MMX technology has been designed to enhance the performance of advanced media and communication applications. Current general-purpose processors have been enhanced with new features explicitly dedicated to the efficient handling of multimedia (audio and video) data. In order to exploit the high intrinsic spatial parallelism of low-level image processing, an SIMD solution has been adopted, processing multiple elements in parallel using a single instruction.

MMX technology provides new features to the Intel processors' family but, at the same time, maintains backward compatibility with all non-MMX operating systems and applications developed for the Intel platform [19,75].

Basically the MMX extensions provide the programmers with the following new features:

- eight new registers (MMX registers);
- four new data types;
- two different arithmetics;
- new instructions.

MMX technology had a strong impact on applications based on the real-time processing of images. For this reason, an MMX-based system was considered as a new alternative platform for automotive applications. In particular, the low-level portion of the processing aimed at the detection of obstacles in the GOLD system was implemented exploiting the MMX features available on the Intel Pentium family processors [74,75]. This new implementation has been integrated into the GOLD system as a software module, which can be alternatively selected instead of the traditional scalar module for the localization of obstacles.

The main characteristics that were considered in this new implementation are:

- the processing of multiple data according to an SIMD computational model (*data parallelism*), and the use of a specific instruction set dedicated to image processing;
- the careful exploitation of the *instruction level parallelism* of the code and the use of the two internal pipelines;

- the reduction of the amount of data in order to fit into the internal processor cache.

Unfortunately, up to now, no compiler is able to handle the new instructions working on MMX registers, nor to automatically perform the parallelization of the code. Therefore, the complete code has been directly written in Assembly language and its performance analyzed and optimized thanks to the use of a Pentium simulator at the hardware level [75]. Nevertheless, this hard and time-consuming task has been rewarded with an enormous processing speed-up: the new MMX-based code runs about 26 times faster than its scalar C version.

5.4.1 *MMX optimization issues*

Major performance gains can be attained with the fastest algorithms and the most efficient data structures, so every code optimization attempt should start at the highest level of abstraction, focusing on more streamlined procedures. Low level code optimization must be used when the algorithms are fully tested, and high-level optimizations can no longer be applied. Since low-level optimizations generally produce code that cannot be easily modified or reused, an efficient implementation of slow code will run slower than an unoptimized implementation of fast code: low-level optimizations can –at most– double the performance of a program, while a better algorithm might make it one order of magnitude faster. MMX coding currently implies the use of Assembly language, therefore very low-level coding techniques are needed. This discussion is focused on strategies that can easily be implemented and guarantee the highest performance payoff.

Data alignment: a misaligned access in the data cache or on the bus costs at least three extra clock cycles on the Pentium processor. Given that an aligned memory access might only take 1 cycle to execute (assuming a L1 cache hit), this is a major performance penalty. An 8-byte datum should be aligned on an 8-byte boundary. Unfortunately, modern operating systems allocate memory on 4 byte boundaries, so that only an average of 50% of all allocated blocks will usually be aligned. This can be fixed by recoding the memory allocator with a user-defined procedure that aligns the allocated block on 8 byte boundary. If the code pushes MMX registers into

Table 5.1 Number of functional units and their accessibility from the U and V-pipes.

Operation	Units	Latency	Throughput	Execution Pipes
ALU	2	1	1	U and V
Multiplexer	1	3	1	U or V
Shift/pack/unpack	1	1	1	U or V
Memory access	1	1	1	U only
Integer register access	1	1	1	U only

the stack, it is necessary to replace the entry and exit code of the procedure to ensure that the stack pointer is aligned too. Some code snippets can be found in [74].

Data alignment is critical when writing memory-bounded MMX code: the execution speed can be boosted by more than 30% by using an efficient memory allocator.

Instruction pairing: the Pentium processor is an advanced superscalar processor: it is built around two general-purpose integer pipelines and a pipelined floating-point unit, allowing the processor to execute two integer instructions simultaneously. The Pentium processor can issue two instructions every clock cycle, one in each pipe. The first logical pipe is referred to as the *U-pipe*, and the second as the *V-pipe*. During decoding of any given instruction, the next two instructions are checked, and, if possible, they are issued such that the first one executes in the U-pipe and the second in the V-pipe. If it is not possible to issue two instructions, then the next instruction is issued to the U-pipe and no instruction is issued to the V-pipe. Therefore the execution of two MMX instructions in a single clock cycle might double the performance of the code by keeping both pipes busy. Unfortunately, this is not always possible: i.e. two shift instructions cannot be allocated on the same clock cycle. In addition not all instructions can be routed to both pipes indifferently: table 5.1 shows the number of functional units and whether they are addressable from the U or the V-pipe.

- Two MMX instructions which both use the MMX shifter unit (pack, unpack, and shift instructions) cannot be paired together since there is only one MMX shifter unit, although they

may be issued in either the U- or V-pipe.

- Two MMX instructions which both use the MMX multiplier unit (`pmull`, `pmulh`, `pmadd` type instructions) cannot be paired together since there is only one MMX multiplier unit. Multiply operations may be issued in either the U- or V-pipe but not in both during the same clock cycle.
- MMX instructions which access either memory or the integer register file can be issued in the U-pipe only. Their scheduling in the V-pipe causes a wait and swap to the U-pipe.

In addition, the traditional dependency rules apply, i.e. the destination register of an instruction issued in a pipe should not match the source or destination register of the instructions issued in the other pipe.

5.4.1.1 *Tuning MMX code: an example*

This section presents an example of a technique, called *Loop Interleaving*, heavily exploited in the optimization of MMX code for Obstacle Detection. MMX code usually consists of a short loop iterated many times over an array of data. Often there is no relationship between two iterations of the same loop, so the output of the second iteration does not depend on the output of the first, and they may be executed in any order (even at the same time). The first step is designing the algorithm so that it occupies the lowest number of MMX registers: there are only 8 of them, so they limit the number of iterations that can be executed at the same time. MMX registers that are shared between the iterations (for example, bit masks) should only be counted once. Having calculated the highest number of iterations that can be executed in parallel (N), the loop can be unrolled by N and then instructions can be moved around to maximize the use of both pipelines.

- MMX instructions that access memory are always executed in the U-pipe, so it is better to spread them around and try to exploit the V-pipe with ALU or shift instructions.
- Two shift instructions cannot be issued in the same cycle, but all shifts of the different iterations will end up very near, because they belong to the same stage of the loop. This prob-

lem can be solved by introducing a slight delay between the iterations, so that the group of MMX instructions that are physically close to each other belong to different stages of their respective iterations. This is best suited to long loops, because in short loops the overhead given by the bunch of instructions belonging to the same iteration (which usually do not pair together) situated at the top of the loop, may offset the benefits of a greater V-pipe exploitation.

5.4.2 Obstacle Detection on an MMX-based processor

A set of MMX routines has been developed that allow the GOLD system to run efficiently on a Pentium MMX; they include all low-level computations, plus some middle level ones. This section describes each low-level filter for the Obstacle Detection functionality together with the analysis of its performance: starting from the difference image computation, arriving at the morphological filter and finally presenting the polar histogram computation.

5.4.2.1 Difference image

The idea is to compute the absolute difference between the left and right image, to reset pixels below a given threshold, then to pack each pixel only using one bit per pixel (eight pixels per byte).

The absolute difference can be easily implemented computing two saturated [74] differences and ORing them:

```
MOVQ    MM2, MM0 ; make a copy of MM0
PSUBUSB MM0, MM1 ; compute saturated difference one way
PSUBUSB MM1, MM2 ; compute saturated difference other way
POR     MM0, MM1 ; OR them together
```

The threshold is loaded into the MM7 register at the startup. Pixels below the threshold are set to zero using a further saturated subtraction, then one bit is extracted from the result, which will only be set if its byte is above the threshold.

Since the image is 128×128 , the use of one bit per pixel shrink the output data down to 2 kbyte allowing a more efficient use of the Pentium's L1 cache memory. Additional benefits of this choice will be shown in the following section. Unfortunately, this packing process requires the implementation

of quite a lot of instructions and an additional phase of bit swapping.

To minimize the performance hit, the *packing process* works on 32-bit word on each loop extracting a bit from each saturated byte (namely a single pixel) and then a *bit swapping* process rearranges two 32-bit words into a single double word (64 bits). The loop interleaving technique has been applied by running two iterations in parallel, and, as a result, dual pipeline exploitation ratio is quite high.

The bit swapping code applies a mask to isolate all the bits with index i , then copies them into position $7-i$ (where i is between 0 and 7). The bit swapping can be highly optimized by only using three MMX registers and, thanks to a round-robin policy, many flow dependencies can be avoided. Careful instruction scheduling is required due to the limitations that exist on the use of shift instructions: only one shift operation can be issued in each clock cycle.

5.4.2.2 Morphological opening filtering

This routine removes isolated pixels from the difference image. The central pixel $\bullet$ is defined as isolated if all $\circ$ pixels are zero as shown in the following structuring element:

$\circ$		$\circ$
$\circ$	$\bullet$	$\circ$
$\circ$		$\circ$

The highly packed image format described in the previous paragraph is now able to give more benefits: this small chunk of data (2 kbyte) fits comfortably into the Pentium MMX's 16 kbyte L1 cache, avoiding slow memory accesses on the system bus. Even more important, the high degree of parallelism allows the processing of 64 pixels for each loop, since each pixel is represented by a single bit. This powerful combination, plus some optimized coding, make this process extremely fast. In order to explain the algorithm, the following situation will be used, where each box represents a binary pixel.

A1	B1	C1	D1
A2	B2	C2	D2
A3	B3	C3	D3

The target is to evaluate the pixels in the second row ($A2 \rightarrow D2$). Applying the rule explained before, B2 is said to be isolated *iff* A1, A2, A3, C1, C2, and C3 are all set to zero. A new row is computed by joining all three rows with the OR operator:

A1+A2+A3	B1+B2+B3	C1+C2+C3	D1+D2+D3
----------	----------	----------	----------

This new row is copied twice and shifted once left and once right by one position:

0	A1+A2+A3	B1+B2+B3	C1+C2+C3
A2	B2	C2	D2
B1+B2+B3	C1+C2+C3	D1+D2+D3	0

All that remains to be done is to join the first and third row with the OR operator, saturate the result and then AND the second row with the resulting row. The B2 cell will contain the B2 value only if at least one pixel amongst A1, A2, A3, C1, C2, and C3 is not zero. This routine can also gain advantage from the Loop Interleaving optimization: the difference image has a width of 128 pixels, which can be packed into two MMX registers. By unrolling the loop by two, a single row in each iteration can be analyzed (actually these two loops are not completely independent, due to the pixels on the edges of the central column that must be exchanged between the loops).

This routine performs very well, with a high instruction pairing ratio and a very good memory access pattern. The difference image's size (2 kbyte) fits perfectly into the L1 cache, therefore most memory accesses do not use the system bus. Given these assumptions, each loop takes 17 CPU cycles to execute, and the whole routine process takes about 2200 cycles. Actually, during the execution of the benchmarks, it took slightly more than 3000 cycles to execute, which is probably due to L2 cache memory access. According to these numbers, each pixel in the difference image is checked and eventually removed in only 0.19 cycles.

5.4.2.3 Polar histogram

This routine builds the polar histogram counting all non-zero pixels on each ray, ranging from 40 to 140 degrees with 1 degree increment and storing the results into an array. Due to the characteristics of the algorithm, it is very difficult to design a parallel algorithm that performs this task, and the simplest and fastest way to solve this problem is to use a lookup table. Therefore, the parallel capabilities of MMX have not been exploited in this routine, nonetheless smart coding speeds up the resulting program compared to non-MMX code: the additional 8 MMX registers help reduce register pressure, whilst MMX shifts are much faster than integer ones. The lookup table is pre-calculated and it is stored with the layout shown in table 5.2 where the number of internal loops is stored as

Number of internal loops (4 bytes)	Padding (4 bytes set to zero)
------------------------------------	-------------------------------

and the internal loop block as shown in table 5.3.

In addition, the Loop Interleaving technique has been applied again to compute 3 pixels in each loop. A higher level of parallelism may give further performance gains, but since the number of pixels in each line will be an integer multiple of the unrolling factor, when using large numbers, the overhead due to padding pixels may offset the benefits of longer loops.

Experimental tests showed that both pipes are fully used most of the time. Some optimizations that may not appear obvious are discussed below:

- Memory accesses must be executed in the U-pipe. Each iteration reads 6 quadwords: these are three groups of bit mask plus image map accesses. The bit mask quad word defines the image map block to be read and the position of the desired bit into the block. This data layout wastes some memory, because only 5 bytes are needed to store this piece of information, but for alignment reasons this number has been rounded up to 8 bytes. The solution of packing all 3-bit positions into a single double word has been considered, but the additional computations offset the reduced number of memory accesses. It can be reasonably assumed that the image map will be stored into the Pentium's L1 cache memory, being small (only 2 kbyte) and recently accessed (it was fully scanned). The position of the bit mask array in the memory hierarchy may vary: on systems

Table 5.2 Layout of the lookup table for polar histogram computation.

Number of internal loops (N_1) of first angle	Internal loop block (repeated N_1 times)	...	Number of internal loops (N_n) of last angle	Internal loop block (repeated N_n times)	End of loop (zero internal loops)
---	--	-----	--	--	-----------------------------------

Table 5.3 Internal loop block for polar histogram computation.

Address of first memory block	First bit to isolate	Address of second memory block	Second bit to isolate	Address of third memory block	Third bit to isolate
-------------------------------	----------------------	--------------------------------	-----------------------	-------------------------------	----------------------

Table 5.4 Execution times of the MMX routines for Average (AL) and Startup (SL) loops at 200 MHz.

	AL (clocks)	%	SL (clocks)	%
Difference image	43177	32.6	83229	36.8
Noise filtering	3054	2.3	4376	1.9
Polar histogram	84867	64.2	136645	60.4
Histogram normalization	463	0.4	970	0.4
Histogram average	716	0.5	1041	0.5
Total	132277		226261	
Execution time (ms)	0.661		1.131	

with a large L2 cache memory it may still be situated there, as a remainder of the previous execution, but it could also be in the main memory. The bit mask array's location is the most important factor when evaluating this routine's performance, as the worst case can bring a 70% speed penalty compared to the best case.

- Memory addressing: the address in the image map is extracted from a MMX register and placed into an integer register to access memory. This process can cause additional penalties: at least one CPU cycle must pass between the writing of the address into the integer register and the reading from that address. This problem was solved by interleaving another memory access that uses a different integer register: by using two kinds of data (bit masks and image map blocks) their memory accesses can be interleaved and

the penalty explained above avoided.

- Integer register access: all instructions that access integer registers must be issued to the U-pipe. This limitation causes two penalties.
- Loop delay: the three iterations are delayed, so that their shift instructions (which cannot be paired together) are located in different zones of the code. There are no penalties caused by shift instructions or register dependencies.

In execution benchmarks this routine took slightly longer than 100,000 cycles to run when the bit mask array was located into the L2 cache, and took slightly less than 170,000 cycles whilst it was lying in the main memory. This difference is striking, and demonstrates that memory optimizations with modern CPU's are much more important than they used to be, as they can offer a big payoff, even better than the ones that can be obtained with coding techniques such as instruction scheduling.

Since the performance bottleneck of the previous algorithm is reading the bit mask table from slow memory, a simple way to improve the running speed is to shrink the bit mask table: two adjacent pixels that belong to the same angle can be encoded in a single table entry, only using one shift index. The bit mask table is then split into two parts: the first one is a list of pixel duos, the second a list of single pixels. In real world benchmarks, this algorithm shrinks the bit mask table to 68,808 bytes reduced from 100,466 bytes, and decreases the execution time by 19%, down to 84,867 cycles.

Each value in the histogram is the number of non-zero pixels that belong to that angle. However, each angle spans a different area, therefore to obtain the ratio between non-zero pixels and all pixels contained in that area, each value of the resulting histogram has to be scaled (the scaling factors are computed at startup). In this case the running time is so short (an average of 463 cycles) that specific optimizations are not worth applying.

Each value contained in the histogram is averaged to smooth spikes. The following formula was implemented (the division by 2 or 4 can be coded as a right shift instruction):

$$\text{Average}(V_i) = \frac{\frac{V_{i-2} + V_{i+2}}{2} + V_{i-1} + V_i + V_{i+1}}{4} \quad (5.2)$$

Also this routine's running time is so short (slightly more than 700 cycles) that any further optimizations are not worth carrying out.

The MMX implementation of the Obstacle Detection functionality fully satisfies real-time requirements, being able to complete all computations in less than one millisecond. Given these numbers, the real bottleneck of the system is the acquisition process, which with traditional CCD cameras takes at least 20 ms. The major performance hit is given by an empty L2 cache, as table 5.4 shows: on the startup loop all data lies in the main memory, while on the average loop it is split between L1 and L2 cache memories. The startup condition is more likely in real world execution, because other routines such as Lane Detection and perspective removal consume a lot of memory, so they probably flush the data out of cache memories.

5.5 Comparison between PAPRICA-3 and MMX-based Processors

This section presents a comparison between the two different platforms PAPRICA-3 and MMX-based processors regarding their use in the processing of images for automotive applications.

5.5.1 Algorithms implementation

The low-level portions of the algorithms described in section 4.3.3 have been implemented and tested on both the PAPRICA-3 and Pentium MMX architectures. The PAPRICA-3 assembly code was generated using the code generator starting from a C++ source, whilst the assembly code for the MMX implementation was written and optimized directly by hand, since up to now there is no standard C compiler that handles MMX extensions. Moreover, in case of future availability, an *automatic parallelizing* compiler would probably not generate a fully optimized code unless some *ad-hoc* C statements are introduced to help the parallelization.

According to the analysis in [5] the two following categories of operations have been considered.

- *Point-wise operations*: the new status of each image pixel is computed as a function of the previous value of the same pixel; they are used for simple operations, such as thresholding or image difference.
- *Cellular Automata operations*: the new status of each pixel is computed as a function of the values of the pixel's neighbors; they are used for operations such as morphological filters or adaptive

thresholding.

For the sake of simplicity and the purpose of comparison in the following two examples—one for each category—are considered: absolute difference between two images (point-wise operation used for Obstacle Detection) and brightness comparison in a horizontal neighborhood (cellular automaton operation used for Lane Detection).

Since these operations are performed on 8-bit grey-level images; the MMX processor handles 8 pixels simultaneously, while PAPRICA-3 operates at the same time on a single bit of every pixel of a full image line.

Absolute difference between two images: a pixel-wise comparison of two images is performed. Every pixel of the resulting image is obtained computing the absolute difference between the corresponding pixels of the two images.

In the case of the MMX-based processor, naming a and b the two images, the following two unsigned differences $ud_1 = a - b$ and $ud_2 = b - a$ are computed. Thanks to the saturation arithmetic, ud_1 is null where $a < b$ as well as $ud_2 = 0$ where $a > b$, therefore the absolute value $ud = |a - b|$ can be computed as $ud = ud_1 \vee ud_2$. Conversely, the PAPRICA-3 implementation is based on a single signed difference ($sd = a - b$) that gives a two's complement 9-bit result, whose ninth bit (s) is the sign. The final result $ud = |a - b|$ is computed as the eight less significant bits of $\overline{sd} + 1$ in case s is set, and as the eight less significant bits of sd otherwise.

Brightness comparison in a horizontal neighborhood: every pixel's value is computed as a function of its value and the values of its left and right neighbors at a distance of 2.

In the MMX implementation, multiple accesses to the memory are required; the two following solutions are possible.

- Using a single access to the memory, the pixel value as well as the values of its neighbors are loaded into a single MMX register; in order to perform the comparison, several shift operations must be used to align the pixels with its second neighbors. Unfortunately, these shift operations reduce the number of pixels that can be processed simultaneously for each memory access from 8 to 4 (figure 5.8.a). Therefore two read accesses to the memory are required for the processing of 8 pixels.

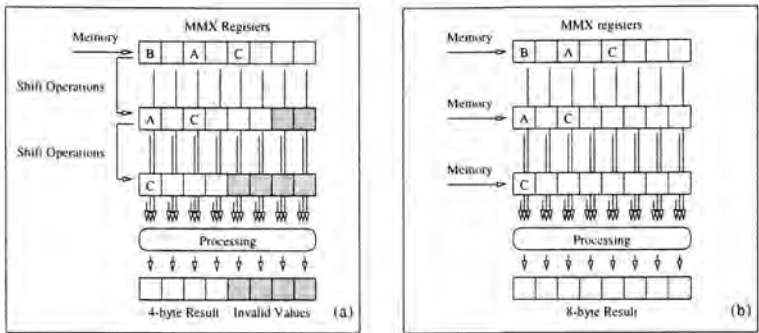


Fig. 5.8 Two different approaches to a cellular automaton operation in the MMX technology involving 8-bit pixels A, B, and C; (a) a single access to the memory followed by shift operations; (b) three partially overlapped accesses to the memory.

- Conversely, using three accesses to the memory, the pixel value and its neighbors' values can be loaded into three different MMX registers so that they have the correct alignment. In this case, 8 pixels can be processed simultaneously and there is no need for shift operations (figure 5.8.b).

On the other hand, in the PAPRICA-3 architecture the PEs can directly access data held in their neighborhood, therefore it is possible to process one full line after a single load operation. Moreover, in the situation where the cellular automaton is based on a wider neighborhood, PAPRICA-3 has no additional overhead, whilst the MMX implementation requires a sufficiently large number of registers and memory accesses.

More generally, when considering point-wise operations on 8-bit deep images, it can be seen that the MMX-based processor and PAPRICA-3 architecture have a similar behavior, the only differences being the instruction sets and the number of pixels that can be processed simultaneously. Nevertheless, since PAPRICA-3 is a single bit processor, it can also handle efficiently images with a generic depth, whilst the MMX instruction set is only able to deal with fixed-depth images (8, 16, 32, or 64 bit/pixel).

Finally, the PEs of PAPRICA-3 are able to directly access the data of the neighboring PEs, therefore allowing a highly efficient implementa-

tion of cellular automata operations. Conversely, in the MMX technology, multiple accesses to the memory are required and this reduces the overall performance, even if this effect is lessened by the presence of multiple levels of cache memory.

5.5.2 Performance evaluation

For comparison purposes, the algorithms described in section 4.3.3 have been evaluated on PAPRICA-3 architecture, featuring 128 processors, a 100 MHz clock and a 20 ns memory access time, and on two different MMX Intel Pentium architectures, featuring a 66 MHz bus speed and a 166 MHz or 200 MHz processor clock.

Table 5.5 Performance evaluation on PAPRICA-3 (128 PEs) and MMX-based architectures.

	Low-level Obstacle Detection	Low-level Lane Detection
PAPRICA-3 (100 MHz)	1.04 ms	1.67 ms
Pentium MMX (166 MHz)	3.2 ms	7.0 ms
Pentium MMX (200 MHz)	2.7 ms	6.2 ms

Table 5.5 shows the results of the low-level portion of Lane and Obstacle Detection algorithms which include both point-wise operations and cellular automata filters. As a first consideration, it is important to note that the processor clock difference between the two MMX-based processors does not seem to equally affect execution time: in fact, despite a frequency increase of 20%, only 11÷15% speedups have been measured. The rationale behind this result is that both systems use the same bus (66 MHz) for memory access which, for this implementation, represents the actual bottleneck.

These results demonstrate that PAPRICA-3 performs from two to four times better than MMX processors depending on the algorithm. It should be considered, however, that these results have been obtained from a configuration where both architectures offer the maximum efficiency: namely in PAPRICA-3's case, when the number of PEs matches the image width, and in the MMX case, when processing 8 bit deep images. While the first condition is assumed to be valid, processing images with a reduced depth is generally envisaged, particularly for real-time constrained applications.

In this case PAPRICA-3 reaches higher efficiency levels than MMX-based processors.

5.5.3 Discussion

The solution based on a special-purpose processing system like PAPRICA-3 requires the complete design of the computer architecture as well as a programming language and a debugging environment. Conversely, the other solution takes advantage of standard development tools and environments but suffers from a less specific instruction set and a less oriented system architecture. Moreover, since the use of MMX extensions relies on an SIMD computational paradigm which cannot be fully exploited by a standard C compiler, specific approaches are needed: in fact, in order to obtain a highly efficient code, either assembly code programming or specific extensions to the C language are required.

Nevertheless, the advantages offered by the first solution, such as an ad-hoc design of both the processing paradigm and the overall system architecture, are diminished by the necessity of managing the complete project, starting from the hardware level (design of the ASICs) as far as the design of the architecture, the programming language along with an optimizing compiler, and finally to the development of applications using the specific computational paradigm.

Furthermore, in comparing these two solutions not only do the architectural details have to be taken into account, but the technological aspects need to be considered as well, the main issues being:

- the advancements of technology, which tend to reduce the life time of the system;
- system design and engineering costs, which are justified for productions based on large volumes only.

These additional considerations highlight that as soon as a new technological process becomes available, the design must be reconsidered from scratch, since new computational paradigms could benefit from the increment in computational power deriving from a higher integration, as it happened for SIMD processors and for MMX extensions.

The development of dedicated architectures was one of the main research topics in the mid 80's, when technology was pushed to its limits and general-purpose processors could not provide sufficient computational

power to support the fast processing of large data structures, like bitmapped images. The general-purpose solution was definitely unsuitable for certain specific problems for which even expensive ad-hoc solutions based on long design, development, and testing times were justified.

Later, when SIMD machines became possible, a large number of simple processing elements were integrated onto the same chip. There was a large explosion of custom architectures, mostly based on processor arrays with a bi-dimensional grid interconnection scheme, since it was thought to be the best solution for image processing. Starting from the original ideas of Unger [153], Fountain [60] classified the different research projects and implementations according to a three generation taxonomy.

The early ILLIAC [16] and CLIP [51] machines belong to the first generation, which were mainly devoted to low-level image processing tasks. The second generation comprises systems such as: the ICL DAP [123], Goodyear MPP [7], and CLIP4 system [61] which evolved towards complete processing systems with dedicated operating systems and languages.

The factor that triggered off the third generation were the availability and widespread use of VLSI technology and the natural mapping of a bi-dimensional interconnection scheme over the planar structure of a silicon chip. This led to an explosion of different proposals and implementations [23, 106, 127, 132, 141, 145] and in some cases to the redesign of previous architectures. Most designs share original characteristics such as the bi-dimensional mesh interconnection scheme and bit serial computation. Instead other ones, such as the Connection Machine, have increased the complexity of the interconnection network or widened the data path of the elementary PE, as in the CLIP7 [62] or MasPar [96] systems. Other machines, such as the AIS [132] and the PAPRICA-3 [22, 68], have a 1-D interconnection scheme emulating a 2D mesh organization.

These architectures demonstrated to be extremely important since they allowed to design and efficiently test solutions based on new approaches and new computational paradigms, such as Cellular Automata [95] or Mathematical Morphology [70, 135].

Moreover, the following availability of fast internetworking facilities gave impulse to the MIMD paradigm, for which processors operating at very high clock frequencies were needed. Thanks to the widespreading of Internet, in the mid 90's multimedia applications gained more importance, and image processing became a basic tool for the solution to problems which were becoming more and more common. Therefore, after a first stage in which

ad-hoc hardware boards were built to support fast image compression –and image processing in general– general-purpose processors were built considering that a lot of their computational power had to be devoted to the processing of images. Although in a period of technological pressure, Intel introduced MMX technology, which, thanks to an overlapping of hardware used in floating point operations and to a small increment of chip size, included specific instructions tailored to the processing of new data structures, such as pixels.

Within this framework, the advantages offered by ad-hoc solutions are more and more confined to a small and specific application field, where hard constraints may impose the design of a specific system architecture, for example in the development of engineered embedded systems.

5.5.3.1 Trends in the automotive field

Similar issues have been addressed by other research institutes working in the automotive field. The great majority of the projects involving the use of machine vision to sense the environment began 5 to 10 years ago with the development of ad-hoc computer architectures that could provide sufficient computational power to support low-level image processing. Among them, the most important and significant examples derive from *Carnegie Mellon University* (section 3.2.8) and *Universität der Bundeswehr* (section 3.2.3), in which both research groups developed their own system architectures: in the first case a 16k Mas-Par MP-2 was installed on the experimental vehicle NavLab I [78, 79], whilst in the second case, special-purpose boards were included in the Transputer-based architecture of VITA.

Nevertheless, both groups are currently heading their research toward the use of general-purpose processors in a commercial architecture with off-the-shelf components, such as frame-grabbers and I/O boards. These systems can be efficiently used to test the algorithms and any new approach without worrying about the advancements of technology during the course of the project. Once the algorithms have been defined, tested, and tuned according to the specific application and environment, the engineering phase may take place in order to produce the final embedded system taking advantage of up-to-date technology.

As a final consideration, current technology allows the possession of sufficiently powerful processing engines to provide an efficient development

infrastructure (considering both hardware devices and programming tools). Obviously, since the final product generally needs specific characteristics such as low cost, minor power consumption, or small physical size, the final phase will consist of the re-engineering the whole system leading to a special-purpose ad-hoc product.

Therefore, although some years ago the extremely expensive development of special-purpose hardware was required because of the low performance of general-purpose systems, the special-purpose solution is nowadays the basis for the final marketing product in specific hard-constrained applications.

This page is intentionally left blank

Chapter 6

The ARGO Vehicle

6.1 Description

ARGO, shown in figure 6.1, is the experimental autonomous vehicle developed at the *Dipartimento di Ingegneria dell'Informazione, Università di Parma, Italy*. It has integrated the main results of the research conducted



Fig. 6.1 The ARGO prototype vehicle.

over the last few years, regarding algorithms and architectures for vision-based automatic road vehicles guidance. Both software and hardware were

first tested and demonstrated on-board MOB-LAB, the mobile laboratory.

Subsequently, thanks to the availability of the ARGO vehicle, a passengers' car equipped with vision systems and an automatic steering capability, a number of different solutions for autonomous navigation has been developed, tested and tuned, particularly for the basic functionalities of Obstacle and Lane Detection. The most promising approaches for both functionalities have been integrated into the GOLD system [10], discussed in section 4.3, which acts currently as the automatic driver of ARGO. Obstacles on the road are detected and localized by means of stereo vision, while the processing of a single monocular image allows the extraction of the road geometry in front of the vehicle.

The web site

<http://millemiglia.ce.unipr.it>

holds up-to-date information and technical data about the ARGO project and its evolution.

6.1.1 The name

The name ARGO derives from the italian name of two distinct myths related to *navigation* and *vision*.

- Argo, the first ship with name in history, used by Jason and a band of 50 mythological heroes to navigate throughout all the Mediterranean sea and to fetch the golden fleece. Since the Argo was equipped with all those implements and tackling necessary for running and guiding it among the huge waves of the open sea, she can represent the archetype of the perfect navigation media.
- Argus, Πανόπτης (the *all-seeing*), the many-eyed god with the gift of all-round vision. Since Argus had a hundred eyes he made a fine watchman, because he could put some of them to sleep whilst the others remained open, and was thus appointed by the goddess Hera to watch over the cow into which Io (Hera's priestess) had been transformed. Disguised as a Shepard, Hermes gained Argus' confidence and lulled him to sleep. Once asleep, Hermes killed Argus. As a memorial, Hera took his eyes and placed them into the tail of her favorite bird, the peacock.

6.2 The Data Acquisition System

Only passive sensors (two cameras and a speedometer) are used on ARGO to sense the surrounding environment: although vision is extremely computationally demanding, it offers the possibility to acquire data in a non-invasive way, namely without altering and polluting the environment. In addition, a button-based control panel has been installed enabling the driver to modify a few driving parameters, select the system functionality, issue commands, and interact with the system.

6.2.1 The vision system

The ARGO vehicle is equipped with a stereoscopic vision system consisting of two synchronized cameras able to acquire pairs of grey level images simultaneously. The installed devices are small (3.2 cm $\times$ 3.2 cm) low cost cameras featuring a 6.0 mm focal length and 360 lines resolution, and can receive the synchronism from an external signal.

The cameras lie inside the vehicle at the top corners of the windscreen, so that the longitudinal distance between the two cameras is maximum. This allows the detection of the third dimension at long distances. The optical axes are parallel and, in order to also handle non flat roads, a small part of the scene over the horizon is also captured, even if the framing of a portion of the sky can be critical due to abrupt changes in image brightness: in case of high contrast, the sensor may acquire over-saturated images.

The images are acquired by a PCI board, which is able to grab three 768 $\times$ 576 pixel images simultaneously. The images are directly stored into the main memory of the host computer thanks to the use of DMA. The acquisition can be performed in real-time, at 25 Hz when using full frames or at 50 Hz in the case of single field acquisition.

6.2.1.1 System calibration

Since the process is based on stereo vision, camera calibration plays a fundamental role in the success of the approach. The calibration process is divided into two steps.

Supervised calibration: the first part of the calibration process is an interactive step: a grid with a known size, shown in figure 6.2, has been painted onto the ground and two stereo images are captured



Fig. 6.2 View of the calibration grid painted on a reference road segment.

and used for the calibration. Thanks to an X Window-based graphical interface a user selects the intersections of the grid lines using a mouse; these intersections represent a small set of points whose world coordinates are known to the system; this mapping is used to compute the calibration parameters. This set of homologous points is used to minimize different cost functions, such as, the distance between each point and its neighbors as well as line parallelism.

This first step is intended to be performed only once when the orientation of the cameras or the vehicle trim has changed. Since the homologous points are few and their coordinates may be affected by human imprecision, this calibration represents only a rough guess of the parameters, and a further process is required.

Automatic parameters tuning: after the supervised phase, the computed calibration parameters have to be refined. Moreover, small changes in the vision system setup or in the vehicle trim require a periodic tuning of the calibration. For this purpose, an automatic procedure has been developed [11]. Since this step is only a refinement, a structured environment, such as the grid, is no longer required and a mere flat road in front of the vision system is sufficient. The parameters' tuning consists of an iterative procedure, based on the application of the IPM transform to stereo images, which takes about 20 seconds: iteratively small deviations from the coarse parameters computed during the previous step are used to

remap the captured images; the aim is to get the remapped images as similar as possible. All the pixels of the remapped images are used to test the correctness of the calibration parameters through a least square difference approach.

6.2.2 The speed sensor

The vehicle is also equipped with a speedometer to detect its velocity. A Hall effect-based device has been chosen due to its simplicity and its reliability and has been interfaced to the computing system via a digital I/O board with event counting facilities. The resolution of the measuring system is about 9 cm/s.

6.2.3 The user interface

Finally, a set of buttons on a control panel allows the user to interact with the system: the driver can select the functionality, modify some driving parameters, and –in this first stage– can use it as an interface for a basic debugging tool. The control panel is shown in figure 6.3.



Fig. 6.3 The control panel, which includes the two sets of leds and a series of buttons.

6.2.4 The keyboard

The on-board PC's keyboard is used as a set of auxiliary buttons to trigger on or off specific debugging facilities and modify the internal driving parameters that are hidden from the driver. It is therefore only used to debug and tune the system's parameters.

6.3 The Processing System

As mentioned in section 5.5, two different architectural solutions were considered and evaluated: a special-purpose and a general-purpose processing system. As previously discussed, although the dedicated architecture presents the advantages of an ad-hoc solution, it requires an expensive and complex design, whilst, on the other hand, the standard architectural solutions offer nowadays a sufficient computational power even for hard real-time constrained applications as this one.

For these reasons, the architectural solution currently installed on the ARGO vehicle is based on a standard 200 MHz MMX Pentium processor. Software performance is boosted exploiting the SIMD MMX enhancements of the traditional instruction set, which permits processing in parallel multiple data elements using a single instruction flow. Figure 6.4 shows the ARGO boot in which the PC is visible.



Fig. 6.4 Internal view of ARGO's boot: both the PC and inverter are visible.

6.4 The Output System

Four different output devices have been installed on ARGO: some of them are actuators, some were integrated in order to test their usefulness, and a few were included just for debugging purposes. Figure 6.9 on page 172 shows an internal view of the driving cabin.



Fig. 6.5 The control panel and on-board monitor.

6.4.1 The acoustical devices

A pair of stereo loudspeakers are used to issue warnings to the driver in case dangerous conditions are detected, e. g. when the distance from the leading vehicle is under a safety threshold or when the vehicle position within the lane is unsafe. The possibility of using the loudspeakers individually is exploited in order to enhance the effectiveness of the acoustical messages. The driver can choose amongst several kinds of acoustic signals: chimes only, vocal messages in different languages (Italian or English) and with a different information content.

6.4.2 The optical devices

Two different optical devices, both shown in figure 6.5, have been installed: a led-based control panel and a color monitor.

The former is used to display the current functionality and some information about the relative position of the vehicle compared to the lane, namely the vehicle offset with respect to the road center line, as shown in figure 6.6.



Fig. 6.6 The output of the GOLD system on the on-board monitor.

Figure 6.7 shows the debugging tool, running under the X11 windowing system; it allows to tune the GOLD parameters and to compute processing statistics.

The latter is mainly used as a debugging tool, since its integration on a commercial vehicle would distract the driver's attention: a visual feedback is supplied to the driver by displaying the results of the process on a 6"

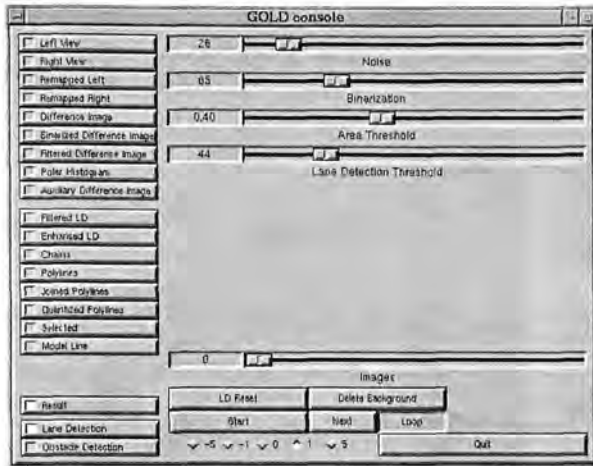


Fig. 6.7 The software control panel used for debugging.

on-board color monitor. The monitor presents the acquired left image with markings highlighting lane markings as well as the position of obstacles. Moreover, it can also display images representing intermediate processing results and the value of some basic parameters for debugging purposes. It also presents a bi-dimensional reconstruction of the road scene, showing the position of obstacles and lane markings.

6.4.3 The mechanical devices

A single mechanical device, shown in figure 6.8, has been integrated on ARGO to provide autonomous steering capabilities. It is composed of an electric engine linked to the steering column by means of a belt.

This device can operate in three different ways:

- as a *warning* device, it is used to vibrate the steering wheel for a short while in order to warn the driver when an unsafe maneuver is undertaken or in correspondence to a momentary reduction in driving capabilities (e. g. drowsiness, wrong security distance...);
- as an *actuating* device, it is used during humanly operated vehicle driving to restore a safe lateral position of the vehicle in case of



Fig. 6.8 View of the electric engine installed on the steering column.

dangerous situations: it steers the vehicle and releases the control of the steering wheel to the driver only when the vehicle is in a safe position within the lane;

- again as an *actuating* device, to drive the vehicle autonomously as follows: the output provided by the GOLD vision system, such as the vehicle lateral offset, the vehicle yaw relative to the road centerline and upcoming road curvature, are combined to determine the lane center at a given distance in front of the vehicle, and the steering wheel is turned to direct the vehicle towards that point ahead of the vehicle.

6.5 The Control System

Once the road geometry ahead of the vehicle has been recovered, a software module is used to compute signals to be issued to the steering wheel actuator. More precisely, starting from the knowledge regarding road geometry and vehicle speed, this module determines the vehicle yaw and computes the position that the vehicle will assume within a given time interval (in the current implementation it is 1.5 s). In case the future vehicle position is not compatible with the requirements (the vehicle should keep central with respect to the driving lane), a rotation is imposed on the steering wheel.

To keep the control system simple, the steering wheel rotation is computed as the result of a temporal average amongst a set of values proportional to the offsets between the future position and the lane center in successive time instants. Despite the delay caused by temporal average, this low-pass filter has the specific advantage of reducing noise caused by possible incorrect results of the vision system, due—for example—to calibration drifts induced by vehicle movements.

Since the processing rate is high (25 frames per second), the refinements of the steering wheel positions occur at a high rate; this eases the filtering of incorrect, vibrating movements caused by noise.

Moreover, an emergency feature has been added to the control system to avoid sudden and therefore dangerous movements of the steering wheel. Each new position—computed as mentioned above—is compared to the current position and processed through a limiting filter, whose threshold depends on vehicle speed. In other words, when the vehicle is moving at a low speed, two consecutive positions of the steering wheel can be quite different; on the other hand, when driving at a high speed, only small steering wheel movements are permitted.

6.5.1 Lane change maneuvers

By means of specific buttons in the control panel, left or right lane change maneuvers can be triggered off. In order to change the lane, the offset between the vehicle's position and the lane marking that the system is tracking is progressively varied until the vehicle reaches the central position in the new lane. This implies the knowledge of the lane width. Then the system locks onto the new lane marking (delimiting the new lane), which is easily localized lying parallel to the previous one at a known distance.

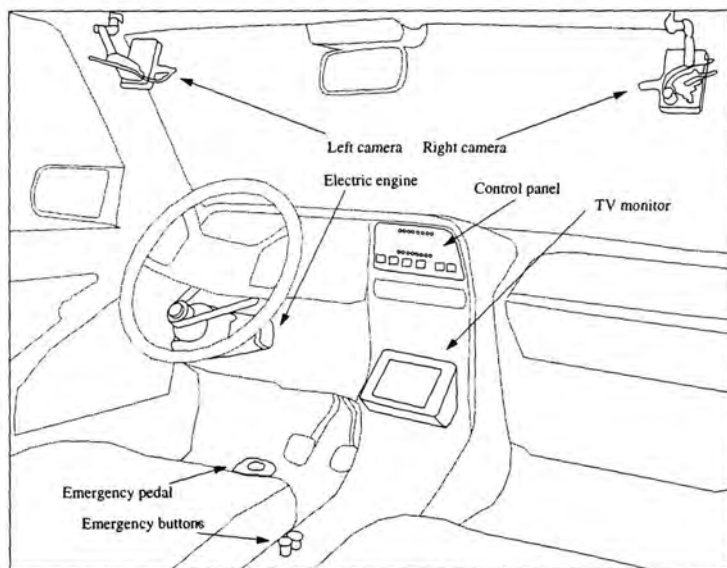


Fig. 6.9 Internal view of the ARGO vehicle.

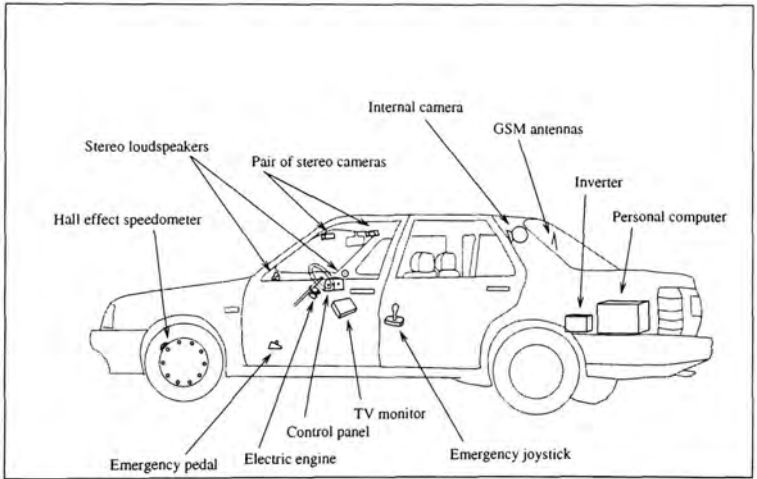


Fig. 6.10 Equipment of the ARGO vehicle.

6.6 Functionalities

Thanks to the control panel, the driver has the possibility of adjusting the values of various driving parameters relative to the driving task and to

select the functionality of the system. The following three functionalities are now integrated.

Manual Driving: the system monitors the driver's activity; in case of potential dangerous situations the system warns the driver with acoustic and optical signals.

Supervised Driving: in case of dangerous situations, the system takes control of the vehicle in order to keep it in a safe position.

Automatic Driving: the system drives automatically, following the lane and localizing obstacles on its path; moreover, it is able to perform lane changes.

6.7 Other Vehicle Equipments and Emergency Features

Besides the main devices for input, output and data processing, some additional equipments have been installed on the vehicle.

- Since all internal instrumentation works at 220 V @ 50 Hz, an inverter has been included to provide the correct power supply.
- The vehicle has been connected to Internet thanks to two GSM cellular channels. The connection is set up on two serial links with the PPP (Point to Point) protocol and the packets are switched onto the two lines for load balancing reasons. This connection permits the transfer of images and statistical data from the vehicle to Internet.
- Finally, the ARGO vehicle has been equipped with several emergency devices to be activated by hand in case of system failures. The availability of these devices has been necessary to ensure safety during the tests and to face insurance requirements. Besides acting on the control panel to deactivate automatic driving features, the passenger can intervene at different levels on a number of emergency devices (see figures 6.9 and 6.10).

Emergency joystick: a joystick is used to overcome the commands issued by the system. The passenger, after pushing one of the two joystick buttons, can take control of the steering wheel by moving the joystick. This emergency feature is intended only to refine the commands issued by the system.

When the button is released, the system keeps working automatically.

Emergency pedal: located near the clutch, it is used to set the *manual driving* functionality. As soon as the pedal is pressed, the control software switches to manual driving and the passenger can take control of the vehicle.

Emergency button: an emergency button, located near the manual brake, performs via hardware the same action of the emergency pedal, detaching the electric engine control signal.

Power supply switch: this last device can be used in case of hardware or software system's failures, and allows the release of the power supply for the steering wheel electric engine.

This page is intentionally left blank

PART III

THE PROJECT'S RESULTS

Downloaded from www.worldscientific.com
by 2601:282:700:6b90:5054:63cf:a8f2:87b6 on 12/22/24. Re-use and distribution is strictly not permitted, except for Open Access articles.

This page is intentionally left blank

This last part of the book presents the results of the ARGO project. The 2000 km test tour is described and the overall system performance discussed. Then, a detailed analysis of the main problems encountered during the whole project is presented for each system component. The final chapter contains some conclusive remarks and a discussion on future trends.

This page is intentionally left blank

Chapter 7

The *MilleMiglia in Automatico* Tour

7.1 Description

In order to extensively test the vehicle under different traffic situations, road environments, and weather conditions, a 2000 km journey was carried out between the 1st and 6th June, 1998. During this test, ARGO drove itself



Fig. 7.1 Automatic driving during the *MilleMiglia in Automatico* tour.

autonomously along the Italian highway network, passing through flat areas and hilly regions including viaducts and tunnels. The Italian road network is particularly suited for such an extensive test since it is characterized by

quickly varying road scenarios which included changing weather conditions and a generally good amount of traffic. The tour took place on highways and freeways, but the system proved to work also on sufficiently structured rural extraurban roads with no intersections.

In the following sections the tour is described, a critical analysis of this test is presented, and the main bottlenecks of the system, as well as some considerations on their possible solutions are highlighted.

7.1.1 Dates and schedule

The *MilleMiglia in Automatico* tour took place along the Italian road network between the 1st and 6th June, 1998: it was subdivided into 7 stages of about 250 km each, and carried out within 6 days.



- Monday, 1st June 1998
9.00–12.00 Parma - Torino 245 km
Mainly flat.
- Tuesday, 2nd June 1998
9.00–11.00 Torino - Milano - Pavia 175 km
Flat; the Milano by-pass taken.
16.00–20.00 Pavia - Milano - Ferrara 340 km
Flat; again through the Milano by-pass.

- Wednesday, 3rd June 1998
12.00–15.30 Ferrara - Bologna - Ancona 260 km
The Bologna by-pass taken; flat until Rimini, then hilly crossing the Appennine region towards Ancona.
- Tuesday, 4th June 1998
8.30–13.00 Ancona - Pescara - Roma 365 km
Hilly until Roma.
- Friday, 5th June 1998
9.00–12.30 Roma - Firenze 280 km
Flat and hilly; the GRA (*Grande Raccordo Anulare*) taken.
- Saturday, 6th June 1998
8.30–12.00 Firenze - Bologna - Parma 195 km
Mountainous: crossing the Appennine region as far as Bologna, then flat until Parma.

7.1.2 Data logging

During the journey, besides the normal tasks of data acquisition and processing for automatic driving, the system logged the most significant data, such as speed, position of the steering wheel, lane changes, user interventions and commands, and dumped the whole status of the system (images included) in correspondence to situations in which the system had difficulties in detecting reliably the road lane.

This data has been processed off-line after the end of the tour in order to compute the overall system performance, such as the percentage of automatic driving, and to analyze unexpected situations. The possibility of re-processing the same images, starting from a given system status, allows the reproduction of conditions in which the fault was detected and a way of solving it is found. At the end of the tour, the system log contained more than 1200 Mbyte of raw data.

7.1.3 Live broadcasting of the event via Internet

During the tour, the ARGO vehicle broadcasted a live video stream on the Internet: two GSM cellular modems were connected to the Vision Laboratory of the *Dipartimento di Ingegneria dell'Informazione* in Parma, and used to transfer both up-to-date news on the test's progression and images acquired by a camera installed in the driving cabin, shown in figure 7.2,



Fig. 7.2 The internal camera –installed over the back seats– framing the driving cabin; a GSM antenna is also visible in the background.

demonstrating automatic driving.

The set-up of a permanent data link from a moving platform implies the use of mobile telecommunications facilities, such as GSM modems. The use of GSM modems for live video streaming requires facing the following constraints:

- low bit-rate for transmission (usually 9600 bps) and
- high bandwidth variability during movements.

In order to increase the throughput of the link, two modems –and therefore two channels– working simultaneously have been installed on ARGO. The transmission hardware is composed of a PC running Linux with two serial ports and a color camera connected to the parallel port. The network traffic is split across the two serial channels thanks to the EQL (Equalizer Load-balancer) [9] protocol, able to split the network traffic across multi-

ple links. Moreover, the communication software was designed in order to dynamically adapt the network traffic to the throughput.

To prove the scientific community, mass-media and general public's high interest, the ARGO web site (<http://mille miglia.ce.unipr.it>) received more than 350,000 contacts during the tour and more than 3000 Mbyte of information were transferred, with a peak of 16,000 contacts per hour during the first day of the tour.

This page is intentionally left blank

Chapter 8

Performance Analysis

This chapter deals with various issues related to system performance. It covers the following: hardware issues regarding the vision system (section 8.1.1), considerations on processing speed (section 8.1.2) and image processing algorithms (section 8.1.3), issues related to the vehicle control system and in particular to the visual feedback used in the vehicle control loop (section 8.1.4), the experience regarding the use of the man-machine interface installed on ARGO (section 8.1.5), and a discussion on how environmental conditions influence the whole system (section 8.1.6). This chapter ends with an overview of some of the problems encountered during the tour and a detailed analysis of one hour of automatic driving.

8.1 System Performance

8.1.1 *The vision system*

Since one of the aims of the whole project is the development of a sufficiently low cost system so as to ease its integration in a large number of vehicles, the use of low-cost acquisition devices was a clear starting point: in particular videophone cameras (small-sized sensors at an average cost of 100 USD each) were installed.

Although having a high sensitivity even in low light conditions (such as during the night), a quick change in the illumination of the scene causes a degradation in the image quality (for example at the entrance or exit from a tunnel). In particular, having a slow automatic gain control (they have been designed for applications characterized by constant illumination such as

video-telephony), in correspondence to the exit from a tunnel for a period of time of about $100 \div 200$ ms, the acquired images result completely saturated and their analysis becomes impossible. This problem also happens when the vehicle crosses an area that contains patches of new (black) and old (light grey or even white) asphalt. A possible solution to this problem could be to use cameras with a higher performance rate and, in particular, a faster automatic gain control and higher dynamics. CMOS-based sensors are also now being evaluated: their logarithmic response should allow higher dynamics, and their intrinsic slowness due to the specific pixel-by-pixel addressing mode, should pose no additional problems since the early stages of processing (IPM) are based on a strong subsampling of the incoming image. Moreover, their use should speed-up the acquisition process since a complete scanning and transfer of the whole image is no longer required.

8.1.2 *The processing system*

The design of the processing system proved to be appropriate for the automatic driving of the vehicle. Moreover, current technology provides processing systems with characteristics that are definitely more powerful than the one installed on ARGO: a commercial PC with a 200 MHz Pentium processor and 32 Mbyte of memory.

On such a system, enhanced by a video frame grabber able to acquire simultaneously 2 stereo images (with 768×576 pixel resolution), the GOLD system processes up to 25 pairs of stereo frames per second and provides the control signals for autonomous steering every 40 ms*. Obviously the processing speed influences the maximum vehicle speed: the higher the processing speed, the higher the maximum vehicle speed.

8.1.3 *The visual processing*

The approach used for both Obstacle and Lane Detection, based on the Inverse Perspective Mapping transform, proved to be effective for the whole trip. Even if on Italian highways the flat road assumption (required by the IPM transform) is not always valid, apart from a exceptions, the approximation of the road surface with a planar surface was acceptable. The calibration of the vision system, in fact, should be tuned to reflect the

*This is equivalent to one refinement on the steering wheel position for every meter when the vehicle drives at 100 km/h.

modifications of road slope ahead of the vehicle, since a wrong calibration generates a lateral offset in the computation of the vehicle trajectory. Nevertheless, since the highway lanes' width is sufficiently large, this offset has never caused serious problems, but it is for this reason that an enhancement to the IPM transform is currently under final testing allowing to get rid of the flat-road assumption and to also handle generic roads.

The only drawback which exists, due to the use of the IPM transform, is that the vehicle movements (pitch and roll) do not allow a reliable detection of obstacles at distances further than 50 meters.

During navigation the system locks onto the right lane marking, hence the vehicles in front do not occlude its visibility. This choice, however, is critical during the complex situations of highway exits, where two lines are present, a continuous one for the exit and a dashed one for the lane. In such cases the passenger can select whether to stay on the road or to exit it.

On the contrary, the resolution of these devices turned out to be satisfactory and the framing of the scene (no horizon to reduce strong light conditions and direct sunlight) correct for the type of roads considered (sloping gently).

During the whole tour the system processed about 1,500,000 images[†] totalling about 330 Gbyte of input data.

8.1.4 The control system

The control system has been designed and developed focusing on its simplicity. Regarding the *mechanical* part, an electric stepping motor allows the rotation of rotate the steering wheel with a high resolution and a reduced power consumption.

On the other hand, the simplicity of the *logical* part of the control system has the main advantage of keeping the entire system robust: only a temporal average and a simple measurement are used to compute the steering wheel angles.

With this kind of control, for speeds reaching around $90 \div 95$ km/h there is no noticeable difference in comparison to a human driver, while for higher speeds (during the test a peak of 123 km/h was reached) the vehicle tends to demonstrate slightly unstable behavior, oscillating inside the lane.

[†]Each with size 768×288 pixel.

A more sophisticated control system is currently under study also including a strong road model; this should allow reaching higher speeds with more stability. This new control mechanism is undergoing tests on a system simulator that permits the emulation of vehicle behavior on curved roads with more than one lane. The simulator has also been used to test lane change maneuver.

8.1.5 *The man-machine interface*

The automatic driving system is managed by a control panel by means of six buttons and eight leds which indicate the currently selected driving functionality. Since this is still in an early development stage, it is possible to also modify the value of a number of driving parameters, which should normally be hardcoded and not adjustable by the final user. Nevertheless, this possibility does not increase the interface complexity. For security reasons the user is required to press two buttons simultaneously for each command to be executed. The modifications in the system's state are then notified to the user through vocal messages which confirm the reception and execution of the command. Moreover, for some commands whose execution is not instantaneous (such as the lane change command) an acoustic signal is also generated upon completion of the maneuver.

The emergency systems allows the driver to take the control of the vehicle during emergency situations in three different ways: tuning driving parameters (through the control panel buttons), temporarily taking over the system (by means of a joystick), completely taking over the system (by pressing a pedal close to the clutch, or the control panel buttons, or an emergency button close to the hand brake).

8.1.6 *Environmental conditions*

During the six days test, the system's behavior was evaluated in various environmental conditions. The route was chosen in order to include areas with different morphological characteristics: from flat areas (Parma, Piacenza, Torino, Milano, Verona, Padova, Ferrara, Bologna), to sloping territories of the Appennines region (Pescara, L'Aquila, Roma, Firenze, Bologna), and heavy traffic zones (Roma, Torino and Milano's by-passes), inevitably encountering stretches of highway with road works, absent or worn horizontal road signs, and diversions.

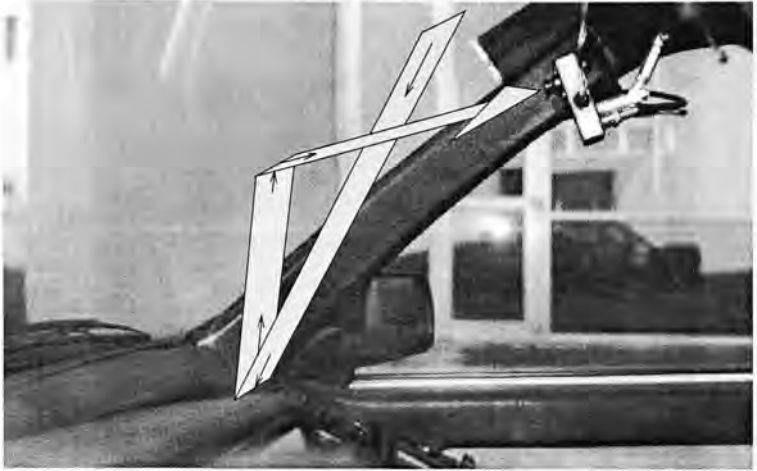


Fig. 8.1 The light's reflection within the internal surface of the windscreen; this causes the images to be over-saturated.

Moreover, the different weather conditions (in particular the light conditions) demonstrated the robustness of both the approach and image processing algorithms. In fact, the system was always able to extract the information for the navigation task even in critical light conditions, with the sun in front of the cameras, high or low on the horizon, during the evening as well as during the day, with high or low contrast.

At night-time the system's behavior is improved by the absence of sun-light reflections and shadows, whilst the area of interest is constantly illuminated by the vehicle headlights.

One of the problems, which is now being solved through shadowing devices and a light polarizing filter, is the light's reflection within the internal surface of the windscreen, as shown in figure 8.1. Figure 8.2 shows some of the typical effects of this reflection.

Finally, the system proved to be surprisingly robust despite the high temperatures measured during the tour: in some cases the external temperature reached 35 °C and the system continued to work reliably even with no air conditioning.



Fig. 8.2 The reflection problem: (a) image with no reflections; (b) image with a strong reflection; (c) over-saturated image due to reflections on the windscreen.

8.1.7 The data transmission system

The transmission of data (both images and textual information) through the GSM network provided by the Telecom Italia Mobile, TIM, cellular company was performed throughout the whole tour. Generally, the route was satisfactorily covered. The main problems were met during the presence of tunnels (excluding the tunnels between Firenze and Bologna, which are almost completely covered) and in the mountainous stretch of highway of the Appennine hills between Pescara and Roma. As an example, figure 8.3 shows the behavior of the GSM connection on the route between Brescia and Ferrara.

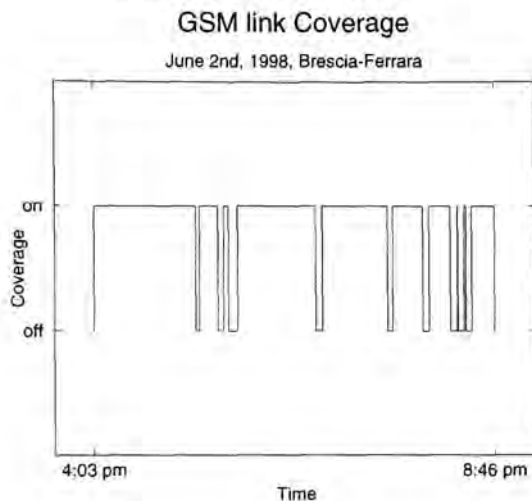


Fig. 8.3 GSM coverage on the route between Brescia and Ferrara.

8.2 Statistical Analysis of the Tour

The analysis of the data collected during the tour allowed the computation of a number of statistics regarding system performance (see table 8.1). In particular, for each stage of the tour the average and the maximum speed of the vehicle during automatic driving were computed. The average speed was strongly influenced by the heavy traffic conditions (especially on Torino and Milano's by-passes) and by the presence of toll stations, junctions, and road works.

The automatic driving percentage and the maximum distance automatically driven show high values despite the presence of many tunnels (particularly during the Appennines routes Ancona-Roma and Firenze-Bologna) and of several stretches of road with absent or worn lane markings (Ferrara-Ancona and Ancona-Roma) or even no lane markings at all (Firenze-Bologna). It is fundamentally important to also note that some stages included passing through toll stations and transiting in by-passes with heavy traffic and frequent queues during which the system had to be switched off.

Table 8.1 Statistical data regarding the system performance during the tour.

Stage	Date	Dept.	Arrival	Maximum distance in automatic [km]				
				Percentage of automatic driving				
				Maximum speed [km/h]				
				Average speed [km/h]				
				Kilometers				
1	1 st June	Parma	Torino	245	86.6	109	93.3	23.4
2	2 nd June	Torino	Pavia	175	80.2	95	85.1	42.2
3	2 nd June	Pavia	Ferrara	340	89.8	115	86.4	54.3
4	3 rd June	Ferrara	Ancona	260	89.8	111	91.1	15.1
5	4 th June	Ancona	Roma	365	88.4	108	91.1	20.8
6	5 th June	Roma	Firenze	280	87.5	110	95.4	30.6
7	6 th June	Firenze	Parma	195	89.0	123	95.1	25.9

8.2.1 Detailed analysis of one hour of automatic driving

Figure 8.4 shows the system behavior during one hour period of time (from 7pm to 8pm on 2nd June 1998) of the tour's third stage (approximately 90 km from Verona to Ferrara). The following items are shown in figure 8.4 from top to bottom.

- The selected functionality: during this period of time the system always worked in automatic mode except in 4 situations indicated with A, B, C, and D, in which:
 - A) ARGO passed through two tunnels (near Vicenza);
 - B) the system at the end of an overtaking maneuver did not recognize the lane markings due to an occlusion;
 - C) and D) the vehicle passed two working areas with no lane markings.

In the first two cases, the system switched back to manual driving automatically and warned the driver; whilst in the other two cases the system was switched off manually. It is important to note that in correspondence to these 4 situations the logged data (apart

System behavior

Third stage: Pavia - Ferrara (June 2nd, 1998)

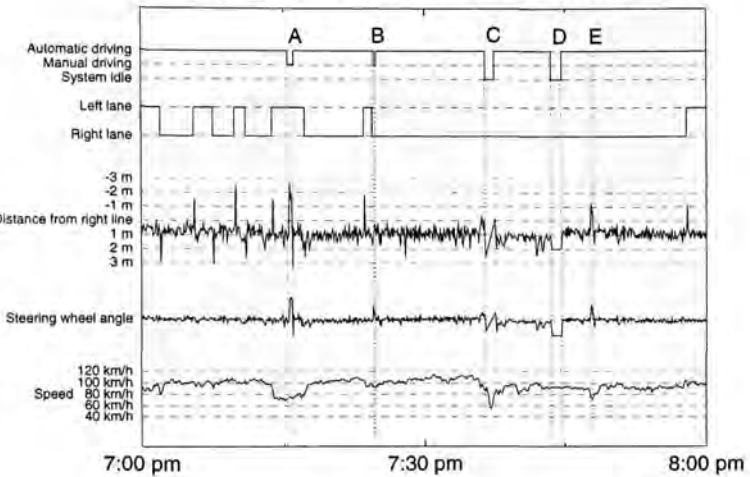


Fig. 8.4 System behavior during one hour of automatic driving.

from speed) is meaningless. The images dumped on disk in these occasions are shown in figure 8.5.

- The vehicle position on the lane: ARGO made a number of lane changes, moving from the rightmost lane to the central lane and vice versa (ARGO was running on a 3-lane highway).
- The distance between the right wheel and the right lane markings (in centimeters). It can be noted that during lane changes the distance alters abruptly; when the vehicle position is again central with respect to the new lane, the systems locks onto the new lane and the distance returns to the nominal value. The selected value of the distance is 80 cm.
- The steering wheel angle automatically chosen by the system. It is possible to note that this angle is proportional to the distance shown by the previous graph, with the only exception of the peaks due to lane changes.



Fig. 8.5 (a), (b), (c), and (d) images dumped in the anomalous situations A, B, C, and D respectively.

- Finally, the speed of the vehicle. It is possible to note the low speed in the situations identified by C and E respectively, due to road works and driving on a junction (between highways A4 and

A13, in which is also visible an evident movement of the steering wheel).

8.3 Discussion

This section summarizes the main problems that lowered the system performance. As mentioned above, the presence of tunnels and long bridges, when combined with strong sunshine, produces very contrasted shadows and sun spots. In order to adjust the value of the automatic gain control, the cameras compute the average brightness of the whole frame. This produces poor results when the scene is highly contrasted as shown in figures 8.6.

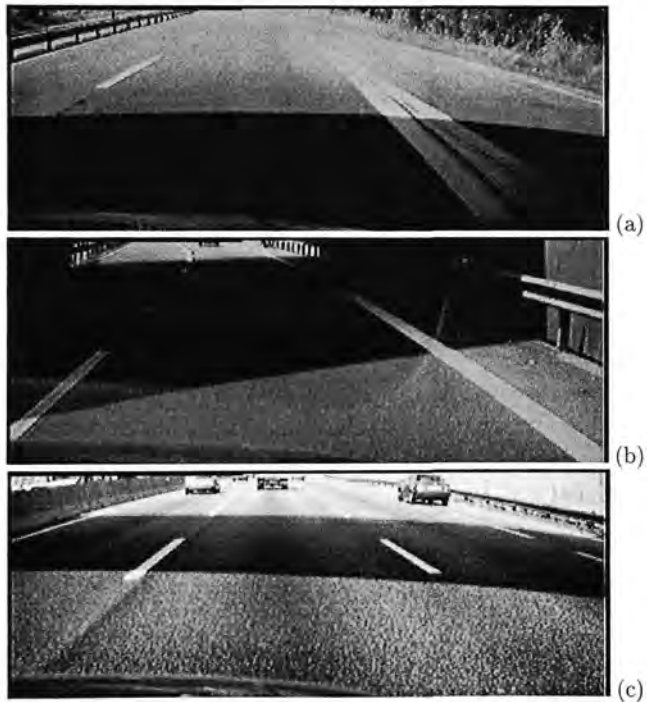


Fig. 8.6 The problem of high contrast: (a) exit from a tunnel; (b) long bridge; (c) short bridge.

Highly contrasted scenes were mainly found in the situations explained above (tunnels and bridges), but also when passing on different patches of asphalt (usually a new dark grey and an old light grey one), as shown in figure 8.7. In the same figure a strong reflection within the internal windscreen surface –which produces a high image saturation– is also visible.



Fig. 8.7 The saturation problem: two different asphalt patches produce a highly contrasted scene which, along with a strong reflection on the windscreen, saturates the image.

Figure 8.8 shows a situation in which the system had a major fault: a vehicle occluded the visibility of the right lane marking in correspondence to a highway exit. Since the only visible lane marking was the continuous line representing the exit, the system followed that line, therefore requiring a user intervention. Since the GOLD system does not include the possibil-



Fig. 8.8 Situation which caused a system fault: a vehicle occluded the visibility of the right lane marking in correspondence to an exit.

ity of automatically choosing whether to stay on the highway or to exit, a user intervention was always requested at each exit. This explains why the percentage of automatic driving was high even when the maximum distance driven in automatic was not that long (see table 8.1). Other user interven-

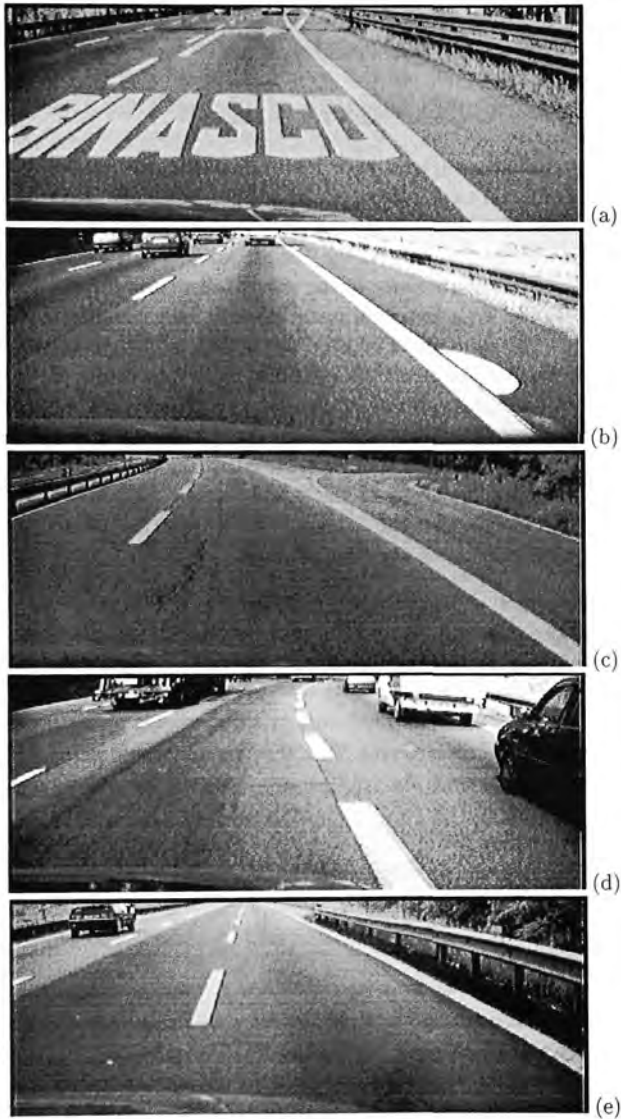


Fig. 8.9 Situations in which the system demonstrated good reliability: (a) horizontal road signs; (b) fog markings; (c) exits; (d) heavy traffic; (e) guard-rail.

tions were required in correspondence to toll stations and road works.

Another serious problem happened in a stretch of road just after the conclusion of road works: on the ground were still present the lines that were painted to delimit the diversion and, as a result, the vehicle followed those lines.

The system demonstrated high robustness with respect to the following: horizontal road signs (figure 8.9.a) and fog markings (figure 8.9.b); in presence of forks, junctions, and exits (figure 8.9.c); heavy traffic conditions that were encountered mainly on the Torino, Milano, and Roma's by-passes (figure 8.9.d); and the presence of the guard-rail (figure 8.9.e).

Moreover, also high temperatures, different light conditions (the third stage of the tour terminated during the night), and high speeds (up to 120 km/h) did not influence the stability and robustness of the whole system, both for hardware and software. The vehicle only experienced a slightly oscillating behavior in correspondence to sloping stretches of road.

Finally, a few minor problems experienced during the tour were: a power failure due to an overloading of the power supply, interference of an on-board cellular phone with the cameras and acquisition board, and some human errors in the use of the control panel.

Chapter 9

Closing Remarks

The experience gained during these years of work within the ARGO project clearly highlighted some of the main problems of automatic driving, whilst the extensive use of the ARGO prototype helped to find their most promising solutions.

The main aims that were established at the beginning of the project were achieved, namely the development of a prototype vehicle and its use as a test-bed for both the hardware and software aspects of the project. The results were extremely rewarding: the *MilleMiglia in Automatico* tour was followed through the Internet by a large number of researchers, press agencies, and people in general worldwide. Thanks to these results, the group of the *Università di Parma* contributed to the research activities on ITS carried out by the international scientific community.

The main aspects of future research within the ARGO project, phase II, are related to the development of a new vehicle integrating both road following and platooning functionalities. The critical analysis of the results of the *MilleMiglia in Automatico* tour as well as the experience gained during the last few years will allow dealing with the main problems encountered, in order to build a more robust prototype.

The topics presented in this book have a strategic importance nowadays, since *active* safety systems are being considered by automotive industries as a basic feature of vehicles of the future. All aspects related to ITS are covered in the series of IEEE Intelligent Vehicles Symposia and IEEE Intelligent Transportation Systems Conferences. To witness this high interest, lately –for the reader's further reference– a number of special issues

have been dedicated to this subject, such as *IEEE Intelligent Systems Magazine* [21], *Engineering Applications of Artificial Intelligence Journal* [20], *Real-Time Imaging Journal* [24], and *Image and Vision Computing Journal* (currently being printed). Furthermore, due to the high impact that these topics will have on life in the future (including both social and economical aspects) a new IEEE Transactions is going to be launched soon and it will be dedicated to ITS topics.

APPENDICES

This page is intentionally left blank

Appendix A

Morphological Implementation of the DBS Filter

In this appendix the rule for the external edge will be considered, assuming step n in the iterative process. Recalling the Mathematical Morphology notations used to identify a grey-tone two-dimensional image, the DT image is a subset of E^3 :

$$DT = \{d \in E^3 \mid d = \langle u, v \rangle, v = V(u), \forall u \in E^2\}, \quad (\text{A.1})$$

where u represents the position of element d in E^2 , and v represents its value.

The *pixelwise masking* operation between a binary and a grey-tone image is defined as a function as follows

$$\odot : E^2 \times E^3 \rightarrow E^3 \quad (\text{A.2})$$

such that $A \odot B$ represents a subset of B containing only the elements $b = (u, v)$ whose position vector $u \in E^2$ also belongs to A :

$$A \odot B \triangleq \{x \in E^3 \mid x = (u, v) \in B, u \in A\} \quad (\text{A.3})$$

In order to compute the the DT 's minimum value in the specified neighborhood, let us consider image $K_e^{(n)}$

$$K_e^{(n)} = \left(S_e^{(n)} \odot DT \right) \cup \left(\overline{S_e^{(n)}} \odot L \right), \quad (\text{A.4})$$

where $S_e^{(n)}$ represents the binary image at step n , the subscript e indicates that the rule for the external edge is being considered, and finally

$$L = \{l \in E^3 \mid l = \langle u, \lambda \rangle, \forall u \in E^2\}. \quad (\text{A.5})$$

to the external edge of $S_e^{(n)}$:

$$E_e^{(n)} = \mathcal{M}(M_e^{(n)}, DT) \cap B_e(S_e^{(n)}) . \quad (\text{A.12})$$

Therefore, the final result of iteration n is given by

$$S_e^{(n+1)} = S_e^{(n)} \cup E_e^{(n)} . \quad (\text{A.13})$$

Figure A.2 shows the execution of an individual iteration of rule 1 on a mono-dimensional image profile.

Following similar steps, it is possible to formalize rule 2. In order to compute the DT 's maximum value in the specified neighborhood, let us consider an image $K_i^{(n)}$

$$K_i^{(n)} = \overline{S_i^{(n)}} \odot DT , \quad (\text{A.14})$$

where the subscript i indicates that the rule for the internal edge is being considered. As shown above, in order to compute the maximum value of a grey-tone image $K_i^{(n)}$ in a 4-connected neighborhood, the following morphological dilation should be used:

$$M_i^{(n)} = K_i^{(n)} \oplus Q , \quad (\text{A.15})$$

where Q is shown in figure A.1.

The set of elements which will be removed from set $S_i^{(n+1)}$ is given by the logical intersection between $\mathcal{M}(M_i^{(n)}, DT)$ and the set of elements belonging to the internal edge of $S_i^{(n)}$:

$$E_i^{(n)} = \mathcal{M}(M_i^{(n)}, DT) \cap B_i(S_i^{(n)}) . \quad (\text{A.16})$$

Therefore, the final result of iteration n is given by

$$S_i^{(n+1)} = \overline{(S_i^{(n)} \cup E_i^{(n)})} = S_i^{(n)} \cap \overline{E_i^{(n)}} . \quad (\text{A.17})$$

Figure A.3 shows the execution of an individual iteration of rule 2 on a mono-dimensional image profile.

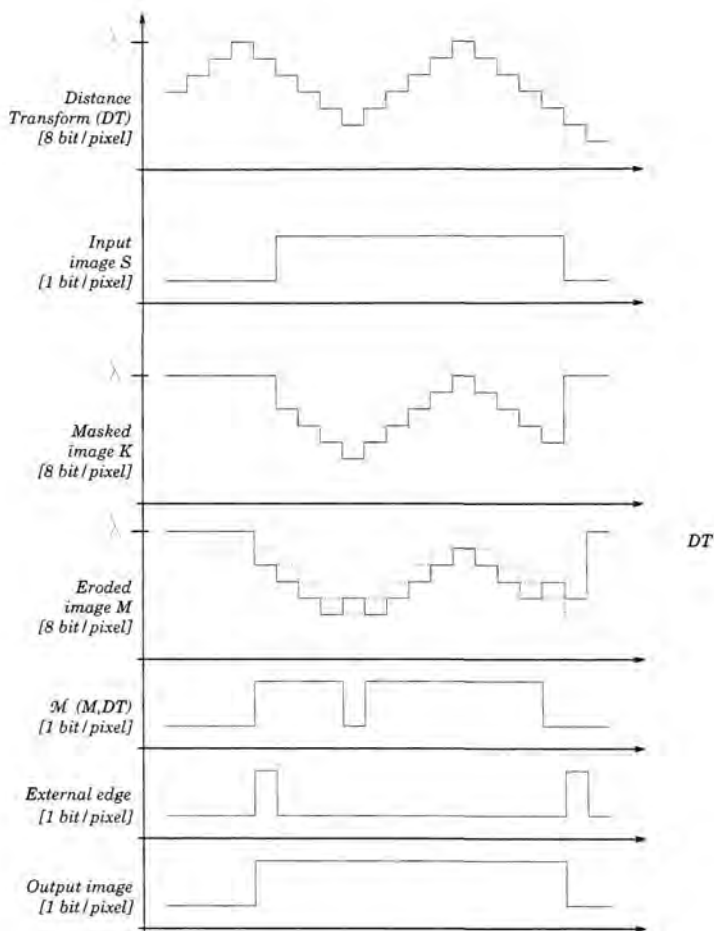


Fig. A.2 Mono-dimensional stretching in the case of an external edge.

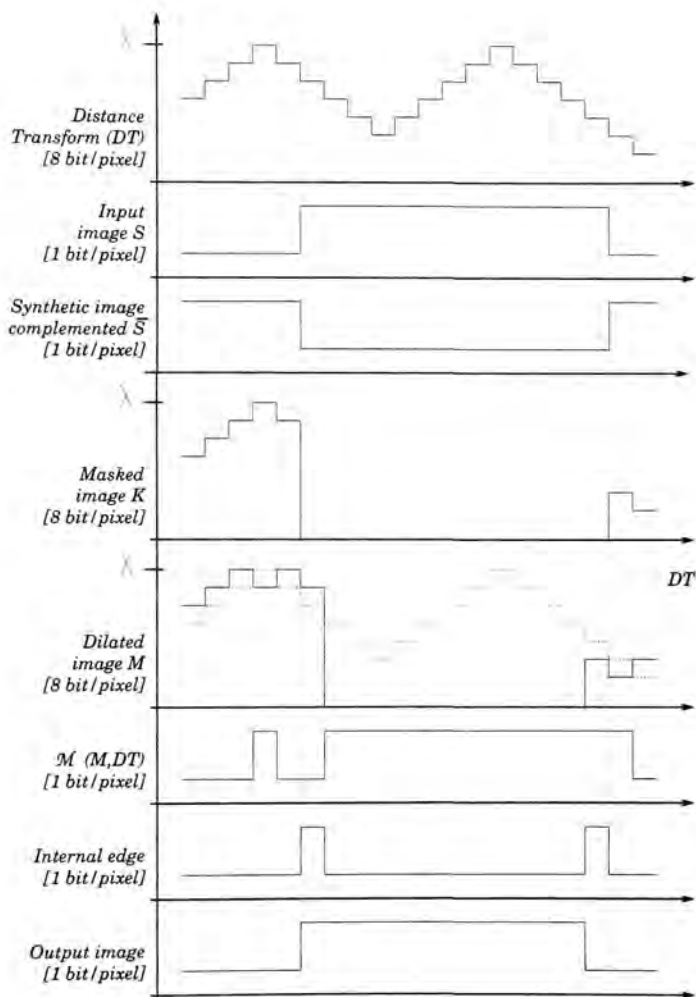


Fig. A.3 Mono-dimensional stretching in the case of an internal edge.

This page is intentionally left blank

Appendix B

PAPRICA-3

Programming Environment

This appendix describes the programming environment developed to ease prototyping of PAPRICA-3 applications. Programs are written in C++; a code generator first converts C++ statements into assembly code; subsequently, optimization is performed on the generated assembly code.

B.1 Low Level Programming Language

Within a generic assembly instruction (called *morphological instruction*) two single-bit operations, a morphological one applied to a given MOR and a logical one applied to the previous result and a second given register (MOR or LOR), can be performed in sequence. The result (1-bit) is placed in a third register:

```
R(25) = MATCH R(12) OR R(7) ;A given match is performed
                                ;on R(12), the result is
                                ;ORed with R(7) and stored
                                ;in R(25)
```

The assembly instructions are coded into fixed-length 32-bit words, where the first 5 bits are reserved for the operation code. Nevertheless, some instructions may need a number of operands whose total length exceeds 27 bits; this is the case of *morphological instructions* (where the complete matching template, the logical operation, the destination, and the two source registers must be specified) and *conditional jumps* (where both the condition and the new absolute memory location must be specified). These instructions are split into two separate ones. In the former, a first

TEMPLATE instruction is used to set the matching template which is used by all the following morphological operations:

```

TEMPLATE -, -, 1, -, -, 1, 1, 1, -, -, 1, -, - ;sets the 13 values
                                           ;corresponding to
                                           ;the enhanced 3x3
                                           ;matching template
R(18) = MATCH R(5)                       ;performs the match on
                                           ;register R(5)

```

In the second case, conditional jumps are split into two adjacent instructions: the former evaluates the condition and sets a flag. In case the flag is evaluated as true, the latter is executed; otherwise the latter is treated as a NOP instruction.

```

ON M > 5                               ;evaluates the condition and sets the flag
  BRC <address>                         ;jumps if the condition was evaluated as true.
                                           ;otherwise BRC is treated as a NOP
                                           ;instruction

```

This solution is particularly efficient for the conditional execution of any other instruction: for example,

```

ON M <= 7                               ;the morphological instruction
  R(6) = MATCH R(5) AND R(8)           ;is executed only when
                                           ;M <= 7

```

This approach also allows easy testing of the combination of two or more conditions, as shown in the following example:

```

ON <Condition1>                         ;test condition 1
  ON <Condition2>                       ;test condition 2
    <INSTR>                             ;execute instruction

```

Instruction INSTR is executed when both Condition1 and Condition2 are evaluated as true and when Condition1 is evaluated as false, namely $(C_1 \cap C_2) \cup \overline{C_1}$.

Although loops can be implemented with conditional jumps, when the number of iterations is known at *assembly time*, the FOR-J construct can be used*. It has been introduced both to improve performance and shorten the code.

*A single counter "J" can be used, therefore no nested constructs are allowed.

```

FOR J=0 TO 7 DO 2           ;Repeat the following <2>
  R(7) = R(7) OR R(9+J)      ;instructions with counter
  R(20+J) = R(9+J) AND R(3) ;J running from 0 to 7

```

PAPRICA-3 features many other assembly instructions. These range from the communication amongst not directly connected PEs, to the direct acquisition and output of images from a camera and to a monitor via a dedicated I/O bus. Nevertheless, the detailed description of these instructions is not presented here since it is not relevant to the following discussion.

B.2 High Level Programming Language

The intrinsic complexity of Assembly language, especially when combined with the SIMD computational model, produces a low programming efficiency. A high-level programming language, together with a software tool used for the generation of its corresponding Assembly code, were needed to ease the development of software applications.

Different solutions were considered, ranging from the development of a completely new ad-hoc language to the use of a code generator. Some considerations led to the choice of a code generator based on the C++ object oriented language: its only drawback is the low-level of optimization achievable in a single pass (generally compilers use two passes).

The main disadvantage of a compiler-based solution lies in the need for a complete definition of a new language, which must be able to handle not only parallel statements, but also some other sequential functions (such as serial operations, data I/O, input of data from the user, memory management, etc.) which are of basic importance although not dealing with the specific SIMD extension. Conversely, a code generator may inherit these capabilities from the chosen high-level language and as a result its implementation is considerably less complex than the former. Moreover, in this second case the efficiency in the development of applications is significantly increased, since the programmer does not have to learn a new language but only its parallel extensions. Furthermore, the use of an object oriented language eases the programmer's task since the operands acting on the new objects can be used intuitively by the application programmer thanks to the possibility of overloading known operators (such as '+', '-', '*', ...).

As shown in figure B.1, the whole sequence of operations required to build a PAPRICA-3 Assembly program is as follows: first the extended

C++ code is written and compiled; then the executable code obtained is run and the PAPRICA-3 Assembly code is generated. The generation of *Assembly code* was preferred with respect to the generation of *binary machine code*, thus requiring a further processing step (PAPRICA-3 Assembler) to get the final binary code. This choice allows a faster and more efficient debugging of the code-generation functions, as well as a simpler implementation of the following global optimization step, since in the Assembly code comments are also automatically included to ease program comprehension. Moreover, since PAPRICA-3 has been conceived as a *real-time* image processor, a code optimizer is needed to further improve the generated code. Finally the PAPRICA-3 Assembler is used to produce the binary machine code.

B.3 Assembly Code Optimization

The optimization process is divided into two main steps: a deterministic one, performed on the initial assembly code, and a stochastic one, iterated on the result of the previous step's result. The first step is used to reduce both execution time and code size and exploits well known traditional optimization techniques [2]; on the other hand, the stochastic step tends to improve the efficiency of the Assembly code exploiting the pipeline structure's characteristics, the specific instruction set, or, more generally, the intrinsic parallelism of each instruction [121]. This process is driven by the user's choice regarding the cost function: execution time, code size, or a given combination of both.

B.3.1 Deterministic optimization

This first step is aimed at applying rules that will certainly lead to shorter and faster versions of the program. The techniques used are: Invariant Code Motion, and Removal of Jumps.

B.3.2 Stochastic optimization

Since in the optimization phase no data is available, data-dependent branches cannot be solved (no *branch prediction* is made here). For this reason, the

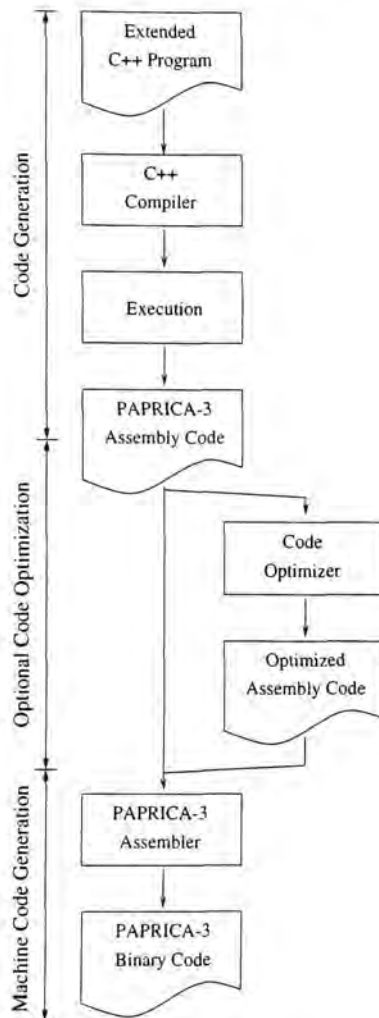


Fig. B.1 The whole sequence of operations required to build a PAPRICA-3 executable.

code is divided into *blocks*, which are defined as code segments without data-dependent conditional control-flow constructs. Each block is optimized independently from the others: a first module of the optimizer tool splits the code into blocks, and sends them to independent processes in charge of the optimization, as shown in figure B.2. Finally the optimized blocks are re-assembled and the final program is generated.

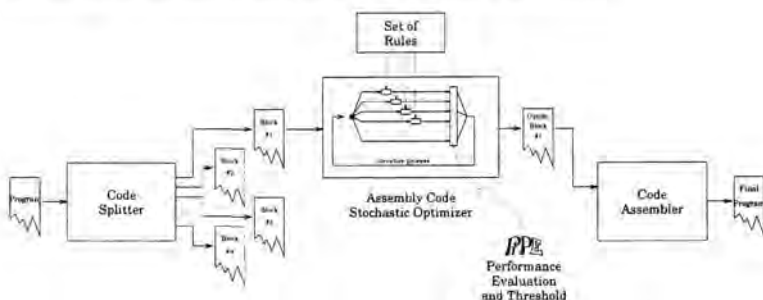


Fig. B.2 Block diagram of the stochastic code optimization.

Each block is optimized following an iterative stochastic approach (as opposed to traditional [45] polynomial solutions [8,17]): at the i -th iteration, $N(i)$ new programs are generated by the consecutive application of $R(i)$ rules (drawn randomly from a given set) to each program of the $(i - 1)$ -th generation. The ones performing better than a given threshold $T(i)$ are kept and will form the i -th generation, while the others are discarded.

Since the population at the first iteration is composed of only the original block, at the beginning of the process (when i is small) the number of rules applied, $R(i)$, is high, therefore allowing a large spreading of the initial population. As soon as i becomes large, the number of rules applied to the same block decreases to 1, for a fine tuning of the solution (see figure B.3.a).

Figure B.3.b shows a qualitative behavior of threshold $T(i)$, which, starting from a value twice as large as the cost featured by the initial program, decreases linearly with i (the slope of $T(i)$ determines the convergence speed). This implies that at the beginning of the iterative process (i small), solutions with a cost higher than the initial can be kept (this is of

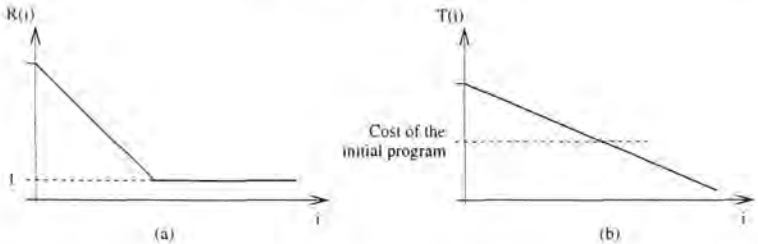


Fig. B.3 Exemplifying behavior of (a) the number of rules applied consecutively to a single block, (b) the threshold $T(i)$ as a function of the iteration number i .

fundamental importance so as to escape from local minima), while, in the following iterations, a threshold limits the number of candidates and only keeps the best solutions. At each iteration the block used to generate the new population is also included in the evaluation process, in order to avoid jittering from the optimal solution that may have been eventually detected. The number of iterations of the stochastic process and the slope $T(i)$ is a function of the length of the block to be optimized: in the first version its dependence is inversely proportional to the iteration number.

The performance of each new program is determined using PiPE, a tool that allows the determination of the performance of a code segment (i.e. the number of clock cycles required for a complete run). Upon initialization, PiPE reads a description of the processor's internal architecture, such as pipeline structure, structural, data, and control hazards [71], as well as the instruction set encoding. Then the list of instructions of a given block is sent to the PiPE tool, which computes and returns the total number of clock cycles required to run that segment of code.

Each rule can move, modify, add, or remove instructions. Every action is fully reversible, thus allowing to return to the original state. The elementary rules used are: Instruction Scheduling, Instruction Alignment, Loop Fusion and Breaking, Loop Unrolling and Rolling, Instruction Fusion and Breaking, Multiple Assignment Removal, Common Subexpression Elimination, and Registers Reallocation.

B.3.3 *Parallel implementation of the code optimizer*

Since the major drawback of a stochastic approach regarding code optimization is the high computational power (in terms of both memory and CPU time), a parallel version of the optimizer has been developed. The parallel version's development began with the analysis of the system's requirements and performance:

- the CPU time spent in the evaluation of the performance of a given segment of assembly code is up to 90% of the overall CPU time;
- the higher the number of iterations, the better the quality of the result, but also the larger the amount of memory required.

Different approaches regarding the development of the optimizer's parallel version were considered (code splitting and assembling is anyway a serial step which is performed by a single master processor). The approach followed in developing the current parallel version (using standard MPI libraries) is based on the distribution of the performance evaluation step only: a master processor splits the code into blocks and generates new blocks to be evaluated. The new blocks are queued and sent to different processors for the evaluation of their performance. When a processor completes the evaluation phase, it returns the number of clock cycles to the master node and signals its availability to evaluate a new code segment. Due to the different computational power required by (i) the evaluation phase (90%) and (ii) the rest of the processing, this approach balances the load even in the case of a non-homogeneous cluster of nodes.

Appendix C

Global Communications on PAPRICA-3

On an SIMD system every PE performs the same operation using the set of data available in its neighborhood; this corresponds to the execution of a Cellular Automata-based operation [42, 56], as shown in figure C.1.a. Figure C.1.b and C.1.c show a set of data transfers between not directly connected cells: these kind of non-local data transfers are called *global communications*. In the first case (figure C.1.b), the distance between source and destination processors is the same for every PE and thus several shift operations allow the performance of all communications simultaneously, using the available PE interconnections. Conversely, figure C.1.c shows a set

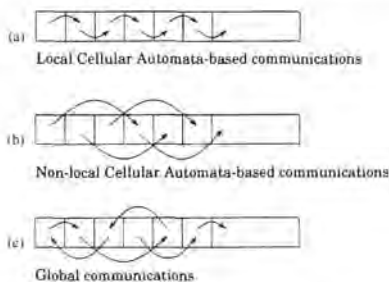


Fig. C.1 Different kinds of interprocessor communications in a mono-dimensional case.

of communications whose source-destination distance is different for each PE. These kind of communications can be efficiently performed by the ICN network, but in general the ICN cannot perform all global commu-

nications simultaneously. As an example, figure C.2.a presents a set of

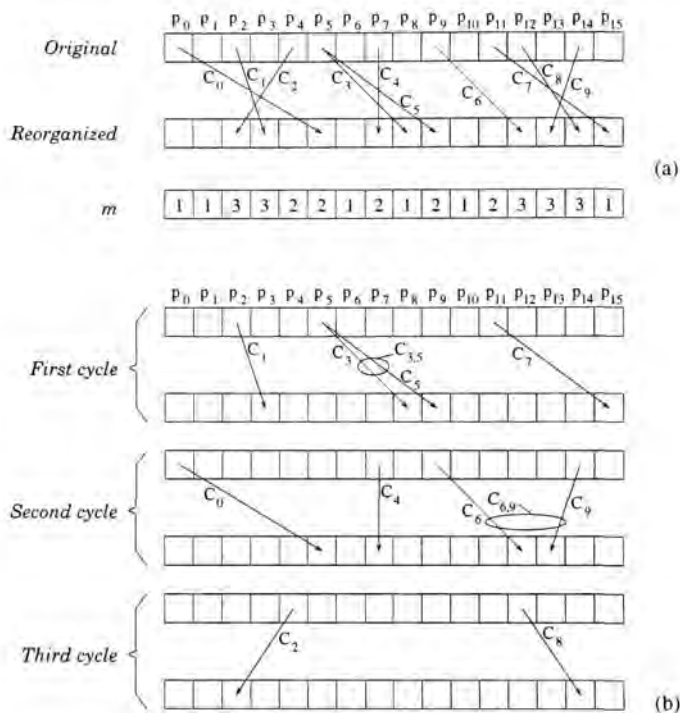


Fig. C.2 Example of line-wise communications on a 16 PE linear architecture.

communications which cannot be performed concurrently since the ICN bus is composed of a single-bit communication line which can be eventually partitioned into independent segments thanks to the set of switches. Each PE can be the source of one or more communications, but obviously it can be the destination of a single communication only. Therefore this process must be serialized in independent steps, each step involving only the communications that can be performed concurrently. An algorithm is therefore required in order to determine the optimal partitioning of the set of communications into the smallest number of subsets. As an example, an optimal partitioning of the set shown in figure C.2.a is depicted in figure C.2.b.

C.1 Concurrent Communications on the ICN

A system composed of a set of N processing elements p_j , with $j = 0, 1, \dots, N - 1$ is considered. Each processor can send or receive data over the ICN communication line, that can be eventually partitioned into independent segments thanks to N switches, each driven by a single processor (see figure 5.3). A communication $C(a, b)$ between processor p_a and p_b , with $a \leq b$ (where p_a is not necessarily the source and p_b the destination), requires the allocation of a portion of the communication line: more precisely the *cluster* of adjacent processors $\{p_a, p_{a+1}, \dots, p_b\}$ must be allocated and uniquely reserved to that communication.

Two communications (or their corresponding clusters of processors) $C(a, b)$ and $C(d, e)$ can be *ordered*, the ordering rules being as follows:

$$\begin{cases} C(a, b) < C(d, e) & \iff a < d; \\ C(a, b) = C(d, e) & \iff a = d; \\ C(a, b) > C(d, e) & \iff a > d. \end{cases} \quad (C.1)$$

Two communications $C(a, b)$ and $C(d, e)$ are said to be *adjacent* iff $d = b + 1$. Two communications $C(a, b)$ and $C(d, e)$ are said to be *compatible* iff there is no overlapping between their two clusters of processors, i.e.:

$$\{p_a, p_{a+1}, \dots, p_b\} \cap \{p_d, p_{d+1}, \dots, p_e\} = \emptyset. \quad (C.2)$$

Therefore, $C(a, b)$ and $C(d, e)$ are *compatible* iff:

$$\begin{cases} b < d & , \text{ when } C(a, b) < C(d, e); \\ e < a & , \text{ when } C(a, b) > C(d, e); \end{cases} \quad (C.3)$$

and cannot be compatible if $C(a, b) = C(d, e)$.

Following these definitions, a *set of communications* is said to be *compatible* if any couple of its elements is also compatible.

Two adjacent communications $C(a, b)$ and $C(b + 1, e)$ are compatible and can therefore be performed simultaneously; in the following discussion, for the sake of simplicity, they will be replaced by a single communication $C(a, e)$. In the example of figure C.2, $C_6 = C(9, 12)$ and $C_9 = C(13, 14)$ are replaced by $C_{6,9}$.

Two communications $C(a, b)$ and $C(d, e)$ sharing the same source processor ($C_3 = C(5, 8)$ and $C_5 = C(5, 9)$ in the example of figure C.2) can

be performed simultaneously, and are therefore replaced by a single communication ($C_{3,5}$) which requires the allocation of the cluster of processors $\{p_{\min(a,d)}, \dots, p_{\max(b,e)}\}$, given by the union of the two original clusters.

Given a set of s communications $C_i \triangleq C(a_i, b_i)$ with $i = 0, 1, \dots, s-1$, each processor p_j is assigned a value $m_j \leq s$ which represents the number of times the allocation of the processor itself is requested by the whole set of communications C_i . The maximum value $w = \max_j m_j$ is called the *multiplicity* of the set of communications C_i . In the example of figure C.2 it is $w = 3$.

It is obvious that a set C_i with multiplicity w cannot be exhaustively partitioned into q non-overlapping (with null intersection) sets of compatible communications, when $q < w$.

Theorem:

Given a communication line with N processing elements p_j , $j = 0, 1, \dots, N-1$, and given any set of s communications C_i , $i = 0, 1, \dots, s-1$, with multiplicity w , it is always possible to partition exhaustively the set of communications C_i into w non-overlapping sets of compatible communications.

Proof:

Let us consider the following set M of processors, defined as:

$$M = \{p_j \mid m_j = w, \text{ with } j = 0, 1, \dots, N-1\}. \quad (\text{C.4})$$

In general, M is formed by $h > 0$ adjacent clusters of processors:

$$M = \left\{ \underbrace{p_{x_0}, p_{x_0+1}, \dots, p_{y_0}}_{\text{Cluster \#1}}, \underbrace{p_{x_1}, p_{x_1+1}, \dots, p_{y_1}}_{\text{Cluster \#2}}, \dots, \underbrace{p_{x_{h-1}}, p_{x_{h-1}+1}, \dots, p_{y_{h-1}}}_{\text{Cluster \#h}} \right\}, \quad (\text{C.5})$$

where $h = \mathcal{C}(M)$ is called *connectiveness* of set M .

Each cluster of processors $T_k = \{p_{x_k}, p_{x_k+1}, \dots, p_{y_k}\}$, with $k = 0, 1, \dots, h-1$, detects a subset H_k of communications C_i such that:

$$H_k = \left\{ C_i = C(a_i, b_i) \mid \{a_i, a_{i+1}, \dots, b_i\} \cap T_k \neq \emptyset, \quad (\text{C.6}) \right.$$

for each $i = 0, 1, \dots, s-1$, with $k = 0, 1, \dots, h-1$.

Since two adjacent communications are handled as a single communication, the allocation of two adjacent processors p_j and p_{j+1} with $m_j = m_{j+1}$ is requested by the same set of communications. As a result, the allocation of processors $\{p_{x_k}, p_{x_k+1}, \dots, p_{y_k}\}$, with $k = 0, 1, \dots, h-1$ of cluster T_k is requested by each one of the elements of H_k .

In the example of figure C.2, it is:

$$\begin{cases} M = \{p_2, p_3, p_{12}, p_{13}, p_{14}\}; \\ h = 2; \\ H_0 = \{C_0, C_1, C_2\}, \text{ and } H_1 = \{C_6, C_7, C_8\}. \end{cases}$$

The proof (by induction) is divided into 3 steps: (a) first the theorem is proven for $h = 1$, (b) then for $h = 2$, and finally (c) assuming it valid for $h = 1, 2, \dots, n-1$ it is proven for $h = n$.

(1) $h = 1$:

in this case, C^1 is formed by just one of the elements of H_0 ; the multiplicity of the remaining set of communications C^2 is then reduced by 1; $\square$

(2) $h = 2$:

when $h = 2$, two different cases arise:

(a) the intersection between H_0 and H_1 is not null;

in this case C^1 is formed by just one of the elements of the intersection between H_0 and H_1 ; the multiplicity of the remaining set of communications C^2 is then reduced by 1. $\square$

(b) the intersection between H_0 and H_1 is null;

in this case H_0 and H_1 represent two disjoint sets of communications, with connectiveness $\mathcal{C}(H_0) = \mathcal{C}(H_1) = 1$, which have been previously considered in step (a) (when $h = 1$). $\square$

(3) $h = n$:

two different cases arise:

(a) the intersection amongst the n sets H_k , with $k = 0, 1, \dots, n-1$ is not null;

in this case C^1 is formed by just one of the elements of the intersection amongst the H_k sets; the multiplicity of the remaining set of communications C^2 is then reduced by 1. $\square$

- (b) the intersection amongst the n sets H_k , with $k = 0, 1, \dots, n-1$ is null;
 in this case the n sets can be partitioned into two disjoint sets H' and H'' , with connectiveness $\mathcal{C}(H')$, $\mathcal{C}(H'') < n$, whose independent partitioning is assumed as demonstrated by the induction process. $\square$

Corollary:

It is always possible to determine a partitioning of any set of s communications C_i with multiplicity w in two disjoint sets where the former is a set of compatible communications (a set with multiplicity 1), and the latter is a set with multiplicity $w - 1$.

Discussion:

- (1) The major result is that the only parameter that influences the communication time is w , which is anyway somewhat dependent on s . However, it does not necessarily follow that an increment in the number of communications s corresponds to an increment in the set multiplicity w . In the worst case (namely when all compatible sets are formed by a single communication) it is $w = s$, but in general it is $w < s$.
- (2) Since the determination of the w sets of compatible communications requires the complete knowledge of the whole set of communications, it can be performed by an external processor, for example a front-end host processor. As a result, this mechanism is extremely efficient in the simulation of pyramidal interconnections between the PEs [27, 129], in the emulation of the non-uniform image sampling of the human visual system [99, 161], or, in general, in the solution of problems involving a fixed set of communications known at compile time, since the determination of the sets of compatible communications is performed off-line. Conversely, its effectiveness decreases in the communication of values whose source and/or destination processors are known only at run-time, since the determination of the sets of compatible communications must be performed at run-time.

C.2 Determining the Sets of Compatible Communications

The algorithm used by the front-end to determine the w sets of compatible communications is directly derived from the corollary discussed above which states that with single scanning it is possible to split the set of s communications (with multiplicity w) into a subset of compatible communications and a set with multiplicity $w - 1$; the problem is thus solved after w iterations of the above procedure. Figure C.3 shows the algorithm used to determine both the multiplicity w of the initial set and the w sets of compatible communications.

<code>count = 0;</code>	Reset the counter of already labeled communications.
<code>w = 0;</code>	Set the initially unknown multiplicity w to 0.
<code>for (i=0; i<s; i++) label[i] = 0;</code>	Mark all communications as unlabeled.
<code>repeat {</code>	Repeat for each subset.
<code>w++;</code>	
<code>for (i=0; i<s; i++) {</code>	Scan all communications.
<code>if ((!label[i]) && (c[i] is compatible with set[w])) {</code>	If the i -th communication has not been labeled yet and if it is compatible with the current w -th set, then label it
<code>label[i] = w;</code>	and increment the counter of already labeled communications.
<code>count++;</code>	
<code>} /* if */</code>	
<code>} /* for */</code>	
<code>until (count==s);</code>	Loop until every communication has been labeled.

Fig. C.3 The algorithm used to determine the w sets of compatible communications.

As explained above, the execution of a single data transfer between a source and as many destinations as required needs the exclusive allocation of a cluster of adjacent PEs. Therefore, for each set of compatible communications, the following three different binary constants (image lines) are defined.

Source: specifying the PEs which act as sources in the current set of communications.

Path: defining the grouping of PEs; since each PE drives a switch that connects itself to its left neighbor, the leftmost PE in a cluster must hold a zero value.

Destination: identifying the destination PEs which must read and store in their local memory the value collected from the ICN network.

The algorithm used to perform the whole set of s communications (w cycles) is shown in figure C.4.

<pre>result = 0; for (i=0; i<w; i++) { s = source[i] & input; commout s, path[i]; commin d; output = d & destination[i]; }</pre>	Reset the final result since each partial result (one for each set of compatible communications) will be accumulated in OR. Loop over the w cycles. Reset all non-source PEs, keeping only one bit for each cluster. Send the value of s over the ICN, configured according to $path$. Store in d the final status read from the ICN network. Accumulate in OR only the values collected from the ICN which correspond to destination PEs.
--	--

Fig. C.4 The SIMD algorithm used to perform multiple global communications over the ICN.

Figure C.5.a and figure C.5.b present an oversampling and an undersampling mapping respectively; in both cases the data movements can be exhaustively partitioned into 3 subsets of compatible communications (requiring 3 cycles of communications).

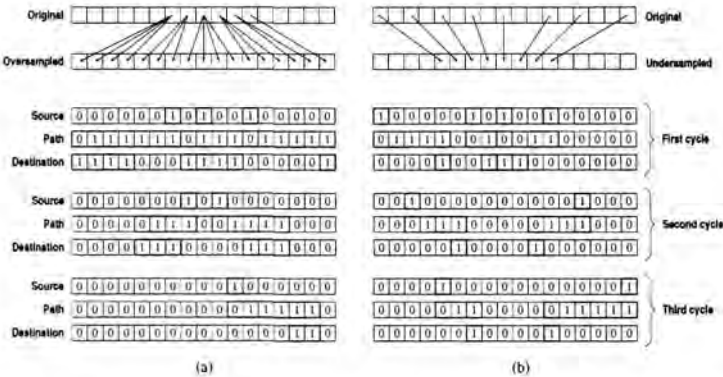


Fig. C.5 Two cases of line-wise global communications used by IPM: (a) oversampling and (b) undersampling.

References

1. G. Adorni, M. Gori, and M. Mordonini. Just-in-Time Sign Recognition in Image Sequences. *Real Time Imaging Journal*, 4, 1998.
2. A. Aho, R. Sethi, and J. Ullman. *Compilers: Principles, Techniques, and Tools*. Addison-Wesley, Reading, MA, 1986.
3. G. Anelli, A. Broggi, and G. Destri. Decomposition of Arbitrarily Shaped Binary Morphological Structuring Elements using Genetic Algorithms. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 20(2):217-224, Feb. 1998.
4. P. C. Antonello, G. Burzio, and G. Vivo. Assistenza alla Guida degli Autoveicoli mediante il Monitoraggio della Corsia Stradale. In Consiglio Nazionale delle Ricerche, editor, *Atti del Terzo Convegno Nazionale Progetto Finalizzato Trasporti 2*, Roma, Italy, Nov. 1997.
5. D. H. Ballard and C. M. Brown. *Computer Vision*. Prentice Hall, 1982.
6. P. H. Batavia, D. A. Pomerleau, and C. E. Thorpe. Overtaking Vehicle Detection using Implicit Optical Flow. In *Procs. IEEE Intl. Conf. on Intelligent Transportation Systems'97*, page 329, Boston, USA, Nov. 1997.
7. K. Batcher. Design of a Massively parallel processor. *IEEE Trans. on Computers*, C-29:836-840, 1980.
8. S. J. Beaty, S. Colcord, and P. H. Sweany. Using Genetic Algorithms to Fine-Tune Instruction-Scheduling Heuristics. In *Procs. IEEE Intl. Conf. on Massively Parallel Computing Systems*, May 1996.
9. M. Beck, H. Bohme, M. Dziadzka, U. Kunitz, R. Magnus, and

- D. Werwornner, *Linux Kernel Internals*. Addison-Wesley, Amsterdam, The Netherlands, second edition, 1998.
10. M. Bertozzi and A. Broggi. GOLD: a Parallel Real-Time Stereo Vision System for Generic Obstacle and Lane Detection. *IEEE Trans. on Image Processing*, 7(1):62–81, Jan. 1998.
 11. M. Bertozzi, A. Broggi, and A. Fascioli. Stereo Inverse Perspective Mapping: Theory and Applications. *Image and Vision Computing Journal*, 1998(16):585–590, 1998.
 12. S. Beucher and M. Bilodeau. Road Segmentation and Obstacle Detection by a Fast Watershed Transform. In *Procs. IEEE Intelligent Vehicles Symposium'94*, pages 296–301, Paris, Oct. 1994.
 13. D. Bishop. Vehicle-Highway Automation Activities in the United State. In *Procs. Intl. AHS Workshop*. U. S. Department of Transportation, 1997.
 14. A. Blake and A. Yuille. *Active Vision*. MIT Press, 1993.
 15. G. Borgefors. Distance Transformations in Digital Images. In *Computer Vision, Graphics and Image Processing*, volume 34, pages 344–371, 1986.
 16. W. L. Bouknight et al. The ILLIAC IV system. *IEEE Procs.*, 60:369–388, 1972.
 17. M. J. Bourke III, P. H. Sweany, and S. J. Beaty. Extending List Scheduling to Consider Extension Frequency. In *Procs. 29th Hawaii Intl. Conf. on System Sciences*, pages 193–202, 1996.
 18. C. Braccini, A. Grattarola, and L. Crovato. Cruise control. In *Procs. 2nd Natl. Conf. of PFT2*, pages 255–256, May 1995.
 19. B. B. Bray. *Programming the 80286, 80386, 80486 and Pentium-based Personal Computer*. Prentice Hall, Inc., Englewood Cliffs, New Jersey, USA, 1996.
 20. A. Broggi. Special-Issue on “Machine Vision for Intelligent Vehicles and Autonomous Robots,” A. Broggi guest-editor. *Intl. Journal on Engineering Applications of Artificial Intelligence*, 11(2):329–330, Apr. 1998.
 21. A. Broggi. Vision-Based Driving Assistance in Vehicles of the Future. *IEEE Intelligent Systems Magazine, Guest Editor's Introduction*, pages 22–23, Nov.–Dec. 1998.
 22. A. Broggi, G. Conte, F. Gregoretti, C. Sansoè, R. Passerone, and L. M. Reyneri. Design and Implementation of the PAPRICA Parallel Architecture. *The Journal of VLSI Signal Processing*, 19(1):5–

- 18, May 1998.
23. A. Broggi, G. Conte, F. Gregoretti, C. Sansoè, and L. M. Reyneri. The Evolution of the PAPRICA System. *Integrated Computer-Aided Engineering Journal - Special Issue on Massively Parallel Computing*, 4(2):114–136, 1997.
24. A. Broggi and F. Gregoretti. Special-Issue on “Special-Purpose Architectures for Real Time Imaging,” A. Broggi and F. Gregoretti guest-editors. *Real Time Imaging Journal*, 2(6):329–330, Dec. 1996.
25. C. Bruedigam. *Intelligente Fahrmanöver sehender autonomer Fahrzeuge im autobahnaehnlicher Umgebung*. PhD thesis, Institut für Systemdynamik und Flugmechanik, Universität des Bundeswehr, 1994.
26. P. J. Burt, P. Anandan, and K. J. Hanna. An electronic front end processor for active vision. In *Procs. SPIE Conf. on Intelligent Robotics*, Nov. 1992.
27. V. Cantoni and M. Ferretti. *Pyramidal Architectures for Computer Vision*. Plenum Press, London, 1993.
28. V. Cantoni, M. Ferretti, and M. Savini. Compact Pyramidal Architectures. In J. Tou and J. Balchen, editors, *Highly Redundant Sensing in Robotic Systems*, volume 58, pages 157–174. NATO ASI Series F, 1990.
29. V. Cantoni and S. Levialdi. *Pyramidal Systems for Computer Vision*. Springer Verlag, Berlin, 1986.
30. S. Carlsson and J. O. Eklundh. Object detection using model-based prediction and motion parallax. In *Procs. European Conf. on Computer Vision*, volume 427, pages 297–306, Antibes, France, Apr. 1990. LNCS, Springer Verlag.
31. A. Chandrakasan, S. Sheng, and R. Brodersen. Low-Power CMOS Digital Design. *IEEE Journal of Solid-State Circuits*, 27(4):473–484, Apr. 1992.
32. P. Charbonnier, F. Diebolt, Y. Guillard, and F. Peyret. Road Marking Recognition using Image Processing. In *Procs. IEEE Intl. Conf. on Intelligent Transportation Systems'97*, page 405, Boston, USA, Nov. 1997.
33. P. Charbonnier, P. Nicolle, Y. Guillard, and J. Charrier. Road Boundaries Detection using Color Saturation. In *Procs. 9th European Signal Processing Con. '98*, Sept. 1998.

34. R. M. H. Cheng, J. W. Xiao, and S. LeQuoc. Neuromorphic Controller for AGV Steering. In *Procs. IEEE Intl. Conf. on Robotics and Automation*, pages 2057-2062, 1992.
35. A. Coda, P. C. Antonello, and B. Peters. Technical and Human Factor Aspects of Automatic Vehicle Control in Emergency Situations. In *Procs. IEEE Intl. Conf. on Intelligent Transportation Systems'97*, 1997.
36. L. D. Cohen. Note on Active Contour Models and Balloons. *CGVIP: Image Understanding*, 53(2):211-218, Mar. 1991.
37. L. D. Cohen and I. Cohen. Finite-Element Methods for Active Contour Models and Balloons for 2-D and 3-D Images. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 15(11):1131-1147, Nov. 1993.
38. B. Courtois. CAD and Testing of ICs and systems: Where are we going? Technical report, TIMA & CMP, Jan. 1993.
39. J. D. Crisman and C. E. Thorpe. UNSCARF, A Color Vision System for the Detection of Unstructured Roads. In *Procs. IEEE Intl. Conf. on Robotics and Automation*, pages 2496-2501, Sacramento, CA, Apr. 1991.
40. J. D. Crisman and C. E. Thorpe. SCARF: A Color Vision System that Tracks Roads and Intersections. *IEEE Trans. on Robotics and Automation*, 9(1):49-58, Feb. 1993.
41. C. Cugiani, L. Giubolini, and R. Tascone. Millimeter-wave guard rail tracking system for the vehicle lateral control. In *Procs. IEEE 4th Intl. Conf. on Vehicle Navigation and Information System*, pages 521-524, Oct. 1993.
42. K. Culik II, L. P. Hurd, and S. Yu. Computation Theoretic Aspects of Cellular Automata. *Physica D*, 45:431-440, 1990.
43. Defence Evaluation & Research Agency. Defence Evaluation & Research Agency Web Site, July 1998. Available at <http://www.dera.gov.uk/>.
44. S. Denasi, C. Lanzone, P. Martinese, G. Pettiti, G. Quaglia, and L. Viglione. Real-time system for road following and obstacle detection. In *Procs. SPIE on Machine Vision Applications, Architectures, and Systems Integration III*, pages 70-79, Oct. 1994.
45. D. J. DeWitt. *A Machine Independent Approach to the Production of Optimal Horizontal Microcode*. PhD thesis, Dept of Computer and Communication Sciences, University of Michigan, Ann Arbor,

- 1976.
46. E. D. Dickmanns. Expectation-Based Multi-Focal Vision for Vehicle Guidance. In *Procs. 8th European Signal Processing Con. '95*, pages 1023–1026, Trieste, Italy, Sept. 1995.
47. E. D. Dickmanns. Knowledge based Real-Time Vision. In *Procs. 2nd Intl. Conf. on Intelligent Autonomous Vehicles*, June 1995.
48. E. D. Dickmans and B. D. Mysliwetz. Recursive 3-D Road and Relative Ego-State Recognition. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 14:199–213, May 1992.
49. F. Diebolt. *Reconnaissance des Marquages Routiers*. PhD thesis, L'Université Louis Pasteur de Strasbourg, Dec. 1996.
50. F. Diebolt. Road markings recognition. In *Procs. 3rd IEEE Intl. Conf. on Image Processing*, page 17P8, 1996.
51. M. Duff, D.M. Watson, T. Fountain, and G. Shaw. A Cellular Logic array for Image Processing. *Pattern Recognition*, 15:229–247, 1973.
52. Z. Duric and A. Rosenfeld. Image Sequence Stabilization in Real-Time. *Real Time Imaging*, 2(5):271–284, Oct. 1996.
53. W. Enkelmann. Obstacle detection by evaluation of optical flow field from image sequences. In *Procs. European Conf. on Computer Vision*, volume 427, pages 134–138, Antibes, France, Apr. 1990. LNCS, Springer Verlag.
54. W. Enkelmann, G. Struck, and J. Geisler. ROMA: A System for Model-Based Analysis of Road Markings. In *Procs. IEEE Intelligent Vehicles Symposium '95*, pages 356–360, Detroit, Sept. 1995.
55. D. Farkas, J. Young, B. Baertlein, and Ü. Özgüner. Forward-Looking Radar Navigation System for 1997 AHS Demonstration. In *Procs. IEEE Intl. Conf. on Intelligent Transportation Systems '97*, pages 672–676, Boston, USA, Nov. 1997.
56. D. Farmer, T. Toffoli, and S. Wolfram, editors. *Cellular Automata*, Amsterdam, Mar. 1983. North-Holland.
57. A. Fascioli. Localizzazione di ostacoli mediante elaborazione di immagini stereoscopiche. Master's thesis, Università degli Studi di Parma - Facoltà di Ingegneria, Italy, 1995.
58. O. Faugeras. *Three-Dimensional Computer Vision: A Geometric Viewpoint*. The MIT Press, cambridge, 1993.
59. G. H. Forman and J. Zahorjan. The Challenge of Mobile Computing. *Computer*, 27(4):38–47, Apr. 1994.
60. T. Fountain. *Processor Arrays: Architectures and applications*.

- Academic-Press, London, 1987.
61. T. Fountain and V. Goetcherian. CLIP4 Parallel Processing System. *IEEE Procs.*, 127E:219–224, 1980.
 62. T. Fountain and K. Matthews. The CLIP 7A Image Processor. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 10(3):310–319, May 1988.
 63. U. Franke, F. Bottiger, Z. Zomotor, and D. Seeberger. Truck Platooning in Mixed Traffic. In *Procs. IEEE Intelligent Vehicles Symposium'95*, pages 1–6, Detroit, 1995.
 64. U. Franke, D. Gavrila, S. Görzig, F. Lindner, F. Paetzold, and C. Wöhler. Autonomous Driving Goes Downtown. In *Procs. IEEE Intelligent Vehicles Symposium'98*, pages 40–48, Stuttgart, Germany, Oct. 1998.
 65. V. Gegenbach, H.-H. Nagel, and F. Heimes. Ein Echtzeitfähiges Bildauswertungssystem zum Einsatz in Autonomen Fahrzeugen und an Robotern. Technical report, Fraunhofer-Institut für Informations und Datenverarbeitung, 1995.
 66. A. Giachetti, M. Campani, R. Sanni, and A. Succi. The Recovery of Optical Flow for Intelligent Cruise Control. In *Procs. IEEE Intelligent Vehicles Symposium'95*, pages 91–95, Paris, Oct. 1994.
 67. V. Graefe and K.-D. Kuhnert. Vision-based Autonomous Road Vehicles. In I. Masaki, editor, *Vision-based Vehicle Guidance*, pages 1–29. Springer Verlag, 1990.
 68. F. Gregoretti, R. Passerone, C. Sansoè, and A. Broggi. The PAPRICA-3 Parallel Processor. In *Procs. MIPRO'96 Symposium*, Opatija, Croatia, May 1996.
 69. A. Hanjalic and B. Girod. 3D motion and scene structure estimation with motion dependent distortion of measurement windows. In *Procs. of IEEE Intl. Conf. on Image Processing*, volume I, pages 61–64, Lausanne, Swiss, 1996.
 70. R. M. Haralick, S. R. Sternberg, and X. Zhuang. Image Analysis Using Mathematical Morphology. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 9(4):532–550, 1987.
 71. J. L. Hennessy and D. A. Patterson. *Computer Architecture: A Quantitative Approach*. Morgan Kaufmann Publishers, Inc., 1990.
 72. W. D. Hillis. *The Connection Machine*. MIT Press, Cambridge, Ma., 1985.
 73. A. Iannio and S. Shapiro. A survey of the hough transform and its

- extensions for curve detection. In *PRIP78*, pages 32–38, 1978.
74. Intel Corporation. *Intel Architecture MMX Technology Developers' Manual*. Intel Corporation, 1997. Available at <http://www.intel.com>.
 75. Intel Corporation. *MMX Technology Programmers Reference Manual*. Intel Corporation, 1997. Available at <http://www.intel.com>.
 76. A. K. Jain. *Fundamentals of Digital Image Processing*. Prentice Hall, 1989.
 77. R. Jain, R. Kasturi, and B. G. Schunck. *Machine Vision*. McGraw-Hill, Inc., 1995.
 78. T. M. Jochem and S. Baluja. Massively Parallel, Adaptive, Color Image Processing for Autonomous Road Following. In H. Kitano, editor, *Massively Parallel Artificial Intelligence*. AIII Publishers in cooperation with MIT Press, 1993.
 79. T. M. Jochem and S. Baluja. A Massively Parallel Road Follower. In M. A. Bayoumi, L. S. Davis, and K. P. Valavanis, editors, *Procs. IEEE Computer Architectures for Machine Perception*, pages 2–12, New Orleans, Dec. 1998.
 80. T. M. Jochem, D. A. Pomerleau, and C. E. Thorpe. MANIAC: A Next Generation Neurally Based Autonomous Road Follower. In *Procs. 3rd Intl. Conf. on Intelligent Autonomous Systems*, Pittsburgh, Pennsylvania, USA, Feb. 1993.
 81. R. E. Kalman. A New Approach to Linear Filtering and Prediction Problems. *Trans. ASME Journal of Basic Engineering*, 82(1):35–45, Mar. 1960.
 82. M. Kass, A. Witkin, and D. Terzopolous. Snakes: Active Contour Models. *Internation Journal of Computer Vision*, 1:321–331, 1987.
 83. K. I. Kim, S. Y. Oh, S. W. Kim, H. Jeong, C. N. Lee, B. S. Kim, and C. S. Kim. An Autonomous Land Vehicle PRV II: Progresses and Performance Enhancement. In *Procs. IEEE Intelligent Vehicles Symposium'95*, pages 264–269, Detroit, Sept. 1995.
 84. K. I. Kim, S. Y. Oh, S. W. Kim, H. Jeong, C. N. Lee, B. S. Kim, and C. S. Kim. An Autonomous Land Vehicle PRV III. In *Procs. IEEE Intelligent Vehicles Symposium'96*, pages 159–164, Tokyo, Japan, Sept. 1996.
 85. K. Kluge. Extracting Road Curvature and Orientation from Image Edge Points without Perceptual Grouping into Features. In *Procs. IEEE Intelligent Vehicles Symposium'95*, pages 109–114, Detroit,

- Oct. 1995.
86. K. Kluge and G. Johnson. Statistical Characterization of the Visual Characteristics of Painted Lane Markings. In *Procs. IEEE Intelligent Vehicles Symposium'95*, pages 488-493, Detroit, Sept. 1995.
 87. K. Kluge and S. Lakshmanan. A Deformable Template Approach to Lane Detection. In *Procs. IEEE Intelligent Vehicles Symposium'95*, pages 54-59, Detroit, Sept. 1995.
 88. K. Kluge and S. Lakshmanan. Lane boundary detection using deformable templates: Effects of image subsampling on detected lane edge. *Lecture Notes in Computer Science*, 1035:329-340, 1996.
 89. K. Kluge and C. E. Thorpe. Intersection Detection in the YARF Road Following System. In *Procs. IEEE Intl. Conf. on Intelligent Autonomous Systems*, pages 145-155, Tokyo, 1993.
 90. D. Koller, J. Malik, Q.-T. Luong, and J. Weber. An integrated stereo-based approach to automatic vehicle guidance. In *Procs. 5th Intl. Conf. on Computer Vision*, pages 12-20, Boston, 1995.
 91. W. Kruger, W. Enkelmann, and S. Rossle. Real-Time Estimation and Tracking of Optical Flow Vectors for Obstacle Detection. In *Procs. IEEE Intelligent Vehicles Symposium'95*, pages 304-309, Detroit, Sept. 1995.
 92. V. F. Leavers and J. F. Boyce. The radon transform and its application to shape parameterization in machine vision. *Image and Vision Computing*, 5:161-166, 1987.
 93. M. Lützel and E. D. Dickmanns. Road Recognition with MarVEye. In *Procs. IEEE Intelligent Vehicles Symposium'98*, pages 341-346, Stuttgart, Oct. 1998.
 94. H. A. Mallot, H. H. Bülthoff, J. J. Little, and S. Bohrer. Inverse perspective mapping simplifies optical flow computation and obstacle detection. *Biological Cybernetics*, 64:177-185, 1991.
 95. N. Margolus and T. Toffoli. Cellular Automata Machines. In G. D. Doolen et al., editors, *Lattice Gas Methods for Partial Differential Equations*, pages 219-249, Redwood City, California, 1990. Addison Wesley.
 96. MasPar Computer Corporation, Sunnyvale, California. *MP-1 Family Data-Parallel Computers*, 1990.
 97. L. Matthies. Stereo vision for planetary rovers: stochastic modeling to near real-time implementation. *Intl. Journal of Computer*

- Vision*, 8:71–91, 1992.
98. M. Mauer, R. Behringer, S. Fuerst, F. Thomanek, and E. D. Dickmanns. VaMoRs-P an Advanced Platform for Visual Autonomous Road Vehicle Guidance. In *Procs. of Intl. Conf. on Mobile Robots IX*, Nov. 1994.
 99. C. A. Mead. Neuromorphic Electronic Systems. *IEEE Procs.*, 78(10):1629–1636, 1982.
 100. M. Meng and A. C. Kak. Fast Vision-guided Mobile Robot Navigation using Neural Networks. In *Procs. IEEE System, Man and Cybernetics '92*, pages 111–116, 1992.
 101. L. E. Mettala and O. Firschein. Reconnaissance, surveillance and target acquisition research for the unmanned ground vehicle program. In *Procs. IEEE Intl. Conf. on Intelligent Autonomous Systems*, pages 257–279, 1993.
 102. S. L. Michael Beuvais, Chris Kreucher. Building World Model for Mobile Platforms using Heterogeneous Sensors Fusion and Temporal Analysis. In *Procs. IEEE Intl. Conf. on Intelligent Transportation Systems '97*, page 101, Boston, USA, Nov. 1997.
 103. M. Mizuno, K. Yamada, T. Nakano, and S. Yamamoto. Robustness of Lane Mark Detection with Wide Dynamic Range Vision Sensor. In *Procs. IEEE Intelligent Vehicles Symposium '95*, pages 171–176, Tokyo, 1995.
 104. C. Morimoto and R. Chellappa. Fast Electronic Digital Image Stabilization For Off-Road Navigation. *Real Time Imaging*, 2(5):285–296, Oct. 1996.
 105. H.-H. Nagel, W. Enkelmann, and G. Struck. FhG-Co-Driver: From Map-Guided Automatic Driving by Machine Vision to a Cooperative Driver Support. *Mathematical and Computing Modeling – Special Issue on Network, Control, Communication, and Computing Technologies for Intelligent Transportation Systems*, 4–7(22):185–212, 1995.
 106. NCR Corporation, Dayton, Ohio. *Geometric Arithmetic Parallel Processor*, 1984.
 107. H. Neumann and H. Stiehl. Toward a computational architecture for monocular preattentive segmentation. In R. G. Hartmann and G. Hauske, editors, *Parallel Processing in Neural Systems and Computers*. Elsevier (North Holland), 1990.
 108. R. Neußer, J. Nijhuis, L. Spaanenburg, B. Höfflinger, U. Franke,

- and H. Fritz. Neurocontrol for lateral vehicle guidance. *IEEE MICRO*, pages 57–66, Feb. 1993.
109. W. M. Newman and R. F. Sproull. *Principles of Interactive Computer Graphics*. McGraw-Hill, Tokyo, 1981.
 110. M. Ohzora, T. Ozaki, S. Sasaki, M. Yoshida, and Y. Hiratsuka. Video-rate image processing system for an autonomous personal vehicle system. In *Procs. IAPR Workshop on Machine Vision and Applications*, pages 389–392, Tokyo, 1990.
 111. J. Oliensis. Shape from Shading as a Partially Well-Constrained Problem. *CVGIP*, 54:163–183, 1991.
 112. M. Otte and H.-H. Nagel. Optical Flow Estimation: Advances and Comparisons. In *Procs. 3rd European Conf. on Computer Vision*. LNCS, Springer Verlag, May 1994.
 113. Ü. Özgüner, B. Baertlein, C. Cavallo, D. Farkas, C. Hatipoğlu, S. Lytle, J. Martin, F. Paynter, K. Redmill, S. Schneider, E. Walton, and J. Young. The OSU DEMO'97 Vehicle. In *Procs. IEEE Intl. Conf. on Intelligent Transportation Systems'97*, page 221, Boston, USA, Nov. 1997.
 114. A. Pentland. Local Shading Analysis. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 6(2):170–187, Feb. 1984.
 115. D. A. Pomerleau. Neural Network Based Autonomous Navigation. In C. E. Thorpe, editor, *Vision and Navigation. The Carnegie Mellon NavLab*, pages 83–93. Kluwer Academic Publishers, 1990.
 116. D. A. Pomerleau. *Neural Network Perception for Mobile Robot Guidance*. Kluwer Academic Publishers, Boston, 1993.
 117. D. A. Pomerleau. RALPH: Rapidly Adapting Lateral Position Handler. In *Procs. IEEE Intelligent Vehicles Symposium'95*, pages 506–511, Detroit, Sept. 1995.
 118. D. A. Pomerleau. Visibility Estimation from a Moving Vehicle using the RALPH Vision System. In *Procs. IEEE Intl. Conf. on Intelligent Transportation Systems'97*, page 403, Boston, USA, Nov. 1997.
 119. D. A. Pomerleau and T. Jochem. Rapidly Adapting Machine Vision for Automated Vehicle Steering. *IEEE Expert*, 11(2), Apr. 1996.
 120. W. K. Pratt. *Digital Image Processing*. John Wiley & Sons, 1978.
 121. B. Ramakrishna Rau and J. A. Fisher. Instruction Level Parallel Processing: History, Overview, and Perspective. *The Journal of*

- Supercomputing*, pages 9–50, 1993.
122. D. Raviv and M. Herman. A new approach to vision and control for road following. In *Procs. Conf. on Computer Vision and Pattern Recognition*, pages 217–225, Hawaii, June 1991.
 123. S. Reddaway. DAP - A Distributed Array Processor. In *1st Annual Symposium on Computer Architectures*, pages 61–65, Florida, 1973.
 124. K. A. Redmill. A Simple Vision System for Lane Keeping. In *Procs. IEEE Intl. Conf. on Intelligent Transportation Systems'97*, page 93, Boston, USA, Nov. 1997.
 125. A. Rieder. Trinocular Divergent Stereo Vision. In *Procs. 13th Conf. on Pattern Recognition*, Aug. 1996.
 126. R. Risack, P. Klausmann, W. Krüger, and W. Enkelmann. Robust Lane Recognition Embedded in a Real-Time Driver Assistance System. In *Procs. IEEE Intelligent Vehicles Symposium'98*, pages 35–40, Stuttgart, Oct. 1998.
 127. I. N. Robinson and W. R. Moore. A parallel processor array architecture and its implementation in silicon. In *Procs. of IEEE Custom Integrated Circuits Conf.*, pages 41–45, New York, 1982. Rochester.
 128. A. Rosenfeld. A characterization of parallel thinning algorithms. *Information Control*, 29:286–291, 1975.
 129. A. Rosenfeld. *Multiresolution Image Processing and Analysis*. Springer Verlag, Berlin, 1984.
 130. B. Ross. A Practical Stereo Vision System. In *Procs. Intl. Conf. on Computer Vision and Pattern Recognition*, pages 148–153, Seattle, June 1993.
 131. Y. Ruichek and J.-G. Postaire. Real-Time Neural Vision for Obstacle Detection Using Linear Cameras. In *Procs. IEEE Intelligent Vehicles Symposium'95*, pages 524–529, Detroit, Sept. 1995.
 132. L. A. Schmitt and S. S. Wilson. The AIS-5000 Parallel Processor. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 10(3):320–330, May 1988.
 133. H. Schneidermann and J. S. Albus. Progress and prospects for vision-based automatic driving. In *Procs. 1993 annual meeting of IVHS AMERICA*, pages 569–575, 1993.
 134. U. Seger, H. G. Graf, and M. E. Landgraf. Vision Assistance in Scenes with Extreme Contrast. *IEEE Micro*, pages 50–56, Feb. 1993.

135. J. Serra. *Image Analysis and Mathematical Morphology*. Academic Press, London, 1982.
136. M. Shoji. *CMOS Digital Circuit Technology*. Prentice Hall, 1988.
137. S. M. Smith. ALTRUISM: Interpretation of three-dimensional information for autonomous vehicle control. *Engineering Applications of Artificial Intelligence Journal*, 8(3):271-280, 1995.
138. S. M. Smith and J. M. Brady. ASSET-2: Real-time motion segmentation and shape tracking. *Trans. on Pattern Analysis and Machine Intelligence*, 17(8):814-829, 1995.
139. I. Sobel. Neighborhood Coding of Binary Images for Fast Contour Following and General Binary Array Processing. *Computer Graphics and Image Processing*, 8(1):127-135, Aug. 1978.
140. K. Storjohann, T. Zielke, H. A. Mallot, and W. von Seelen. Visual Obstacle Detection for Automatically Guided Vehicle. In *Procs. of IEEE Intl. Conf. on Robotics and Automation*, volume II, pages 761-766, 1990.
141. T. Sudo, T. Nakashima, M. Aoki, and T. Kondo. An LSI adaptive array processor. In *Procs. IEEE Intl. Solid-State Circuits Conf.*, pages 122,123,307, 1982.
142. B. Super and A. Bovik. Shape from Texture Using Local Spectral Moments. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 17(4):333-343, Apr. 1995.
143. S. L. Tanimoto and K. Kilger. *Structured Computer Vision: Machine Perception through Hierarchical Computation Structures*. Academic Press, NY, 1980.
144. S. L. Tanimoto, T. J. Ligocki, and R. Ling. A Prototype Pyramid Machine for Hierarchical Cellular Logic. In *Parallel Computer Vision*. Academic Press, London, 1987.
145. Thinking Machines Corporation, Cambridge, Ma. *Connection Machine CM-200 Series - Technical Summary*, 1991.
146. F. Thomanek, E. D. Dickmanns, and D. Dickmanns. Multiple Object Recognition and Scene Interpretation for Autonomous Road Vehicle Guidance. In *Procs. IEEE Intelligent Vehicles Symposium'94*, pages 231-236, Paris, Oct. 1994.
147. C. E. Thorpe, M. H. Hebert, T. Kanade, and S. A. Shafer. Vision and navigation for the carnegie-mellon navlab. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 10(3):362-373, May 1988.

148. C. E. Thorpe, M. Herbert, T. Kanade, and S. Shafer. The New Generation System for the CMU NavLab. In I. Masaki, editor, *Vision-based Vehicle Guidance*, pages 83–110. Springer-Verlag, 1991.
149. M. Tistarelli. Vision-based overtaking control to aid car driving. In *Procs. 2nd Natl. Conf. of PFT2*, pages 2539–2549, May 1995.
150. H. Tokuyama. Asia-Pacific Projects Status and Plans. In *Procs. Intl. AHS Workshop*. U. S. Department of Transportation, 1997.
151. M. A. Turk, D. G. Morgenthaler, K. D. Gremban, and M. Marra. VITS - A Vision System for Autonomous Land Vehicle Navigation. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 10(3), May 1988.
152. B. Ulmer. VITA II - Active Collision Avoidance in Real Traffic. In *Procs. IEEE Intelligent Vehicles Symposium'94*, Paris, 1994.
153. S. Unger. A computer oriented toward spatial problems. *Procs. IRE*, 46:1744–1750, 1958.
154. Y. Wan, F. Cabestaing, and J. Burie. A New Edge Detector for Obstacle Detection with a Linear Stereo Vision System. In *Procs. IEEE Intelligent Vehicles Symposium'95*, pages 130–135, Detroit, Sept. 1995.
155. J. Weber, D. Koller, Q.-T. Luong, and J. Malik. New results in stereo-based automatic vehicle guidance. In *Procs. IEEE Intelligent Vehicles Symposium'95*, pages 530–535, Detroit, 1995.
156. K. P. Wershofen and V. Graefe. A Real-Time Multiple Lane Tracker for an Autonomous Road Vehicle. In *Procs. EURISCO '91*, Corfu, June 1991.
157. W. Whittaker. *Terregator - Terrestrial Navigator*. Carnegie-Mellon Robotics Institute, 1984.
158. D. Willersinn and W. Enkelmann. Robust Obstacle Detection and Tracking by Motion Analysis. In *Procs. IEEE Intl. Conf. on Intelligent Transportation Systems'97*, page 325, Boston, USA, Nov. 1997.
159. J. M. Wolfe and K. R. Cave. Deploying visual attention: the guided model. In *AI and the eye*, pages 79–103. A. Blake and T. Troscianko, 1990.
160. Y. S. Yao and R. Chellappa. Selective Stabilization of Images Acquired by Unmanned Ground Vehicles. *IEEE Trans. on Robotics and Automation*, 13(5):693–708, Oct. 1997.

161. B. Zavidovique and P. Fiorini. A Control View to Vision Architectures. In V. Cantoni, editor, *Human and Machine Vision: Analogies and Divergencies*, pages 13–56, Plenum Press, 1994.
162. Y. Zheng, D. G. Jones, S. A. Billings, J. E. W. Mayhew, and J. P. Frisby. SWITCHER: a stereo algorithm for ground plane obstacle detection. *Image and Vision Computing*, 8:57–62, 1990.
163. W. Ziegler, U. Franke, R. Renner, and A. Kühnle. Computer Vision on the Road: A Lane Departure and Drowsy Driver Warning System. In *Procs. 4th Intl. Conf. on Mobility Technology*, Brasil, Oct. 1995.
164. T. Zielke, M. Brauckmann, and W. von Seelen. Intensity and Edge-based Symmetry Detection with an Application to Car-Following. *CVGIP: Image Understanding*, 58:177–190, 1993.

Biographic Notes



Alberto Broggi received both the Dr.Eng. (Master) degree in Electronic Engineering (1990) and PhD degree in Information Technology (1994) from the Università di Parma, Italy. From 1994 to 1998 he was Full Researcher at the Dipartimento di Ingegneria dell'Informazione, Università di Parma, Italy. From 1998 he is Associate Professor of Artificial Intelligence at the Dipartimento di Informatica e Sistemistica, Università di Pavia, Italy. His research interests include real-time computer vision approaches for the navigation of unmanned vehicles, and the development of low-cost computer systems to be used on autonomous agents. He is

the coordinator of the ARGO project, with the aim of designing, developing and testing the ARGO autonomous prototype vehicle, equipped with special active safety features and enhanced driving capabilities. Prof. Broggi is the author of more than 100 refereed publications in international journals, book chapters, and conference proceedings. Actively involved in the organization of scientific events, Prof. Broggi is on the Editorial Board and Program Committee of many international journals and conferences and has been invited to act as guest-editor of journals and magazines theme issues on topics related to Intelligent Vehicles, Computer Vision Application, and Computer Architectures for Real-Time Image Processing. Prof. Broggi is the Newsletter Editor and Member of the Conference and Publication Committees of the IEEE Intelligent Transportation Systems Council and will be the Program Chair of the next IEEE Intelligent Vehicles Symposium, Detroit, 2000.

broggi@CE.UniPR.IT - <http://WWW.CE.UniPR.IT/people/broggi>



Massimo Bertozzi received the Dr.Eng. (Master) degree in Electronic Engineering (1994) from the Università di Parma, Italy, discussing a Master Thesis about the implementation of Simulation of Petri Nets on the CM-2 Massive Parallel Architecture. From 1994 to 1997 he was a PhD student in information technology at the Dipartimento di Ingegneria dell'Informazione, Università di Parma, where he chaired the local IEEE student branch. During this period his research interests focused mainly on the application of image processing to real-time systems and to vehicle guidance, the optimization of machine code at assembly level, and parallel

and distributed computing. After obtaining the PhD degree, since 1997, he has been holding a permanent position at the same department. He is member of IEEE and IAPR.

bertozzi@CE.UniPR.IT - <http://WWW.CE.UniPR.IT/people/bertozzi>



Alessandra Fascioli received the Dr.Eng. (Master) degree in Electronic Engineering (1996) from the Università di Parma, Italy, discussing a Master Thesis on stereo vision-based obstacle localization in automotive environments. Currently she is a PhD candidate in Information Technology at the Dipartimento di Ingegneria dell'Informazione, Università di Parma. Her research interests focus on real-time computer vision for automatic vehicle guidance and image processing techniques based on the Mathematical Morphology computational model. She is member of IEEE and IAPR.

fascal@CE.UniPR.IT - <http://WWW.CE.UniPR.IT/people/fascal>



Gianni Conte received the Dr.Eng. (Master) degree in Electronic Engineering (1970) from the Politecnico di Torino, Italy. From 1972 to 1986 he was Researcher at the Dipartimento di Elettronica, Politecnico di Torino. From 1986 to 1989 he was Full Professor of Computer Science at the Università di Salerno, Italy. From 1989 he is Full Professor of Computer Science at the Dipartimento di Ingegneria dell'Informazione, Università di Parma, Italy, where he teaches computer architectures and parallel processing and is presently Head of the department. He is author of over 90 technical papers in refereed journals and conferences as well as three books on parallel computer architectures and performance modelling. His research interests include parallel processing architectures, performance evaluation of distributed systems, stochastic Petri net modelling, and VLSI system architectures. Prof. Conte acted also as coordinator of different research activities in the frame of CNR and EU-REKA project. He was coordinator of the Italian PRO-CHIP activities in the frame of the PROMETHEUS project. He is member of IEEE.

conte@CE.UniPR.IT - <http://WWW.CE.UniPR.IT/people/conte>

3986 hc



<http://www.worldscientific.com/>