

Jean-Louis Ferrier
Alain Bernard
Oleg Gusikhin
Kurosh Madani *Editors*

Informatics in Control, Automation and Robotics

9th International Conference, ICINCO
2012 Rome, Italy, July 28–31, 2012
Revised Selected Papers

Lecture Notes in Electrical Engineering

Volume 283

Board of Series Editors

Leopoldo Angrisani, Napoli, Italy
Marco Arteaga, Coyoacán, México
Samarjit Chakraborty, München, Germany
Jiming Chen, Hangzhou, P.R. China
Tan Kay Chen, Singapore, Singapore
Rüdiger Dillmann, Karlsruhe, Germany
Gianluigi Ferrari, Parma, Italy
Manuel Ferre, Madrid, Spain
Sandra Hirche, München, Germany
Faryar Jabbari, Irvine, USA
Janusz Kacprzyk, Warsaw, Poland
Alaa Khamis, New Cairo City, Egypt
Torsten Kroeger, Stanford, USA
Tan Cher Ming, Singapore, Singapore
Wolfgang Minker, Ulm, Germany
Pradeep Misra, Dayton, USA
Sebastian Möller, Berlin, Germany
Subhas Mukhopadhyay, Palmerston, New Zealand
Cun-Zheng Ning, Tempe, USA
Toyoaki Nishida, Sakyo-ku, Japan
Federica Pascucci, Roma, Italy
Tariq Samad, Minneapolis, USA
Gan Woon Seng, Nanyang Avenue, Singapore
Germano Veiga, Porto, Portugal
Junjie James Zhang, Charlotte, USA

For further volumes:

<http://www.springer.com/series/7818>

About this Series

“Lecture Notes in Electrical Engineering (LNEE)” is a book series which reports the latest research and developments in Electrical Engineering, namely:

- Communication, Networks, and Information Theory
- Computer Engineering
- Signal, Image, Speech and Information Processing
- Circuits and Systems
- Bioengineering

LNEE publishes authored monographs and contributed volumes which present cutting edge research information as well as new perspectives on classical fields, while maintaining Springer’s high standards of academic excellence. Also considered for publication are lecture materials, proceedings, and other related materials of exceptionally high quality and interest. The subject matter should be original and timely, reporting the latest research and developments in all areas of electrical engineering.

The audience for the books in LNEE consists of advanced level students, researchers, and industry professionals working at the forefront of their fields. Much like Springer’s other Lecture Notes series, LNEE will be distributed through Springer’s print and electronic publishing channels.

Jean-Louis Ferrier · Alain Bernard
Oleg Gusikhin · Kurosh Madani
Editors

Informatics in Control, Automation and Robotics

9th International Conference, ICINCO 2012
Rome, Italy, July 28–31, 2012
Revised Selected Papers

 Springer

Editors

Jean-Louis Ferrier
Laboratoire d'Ingénierie des Systèmes
Automatisés (LISA)
Institute of Science and Technology
of Angers Engineer (ISTIA)
Angers
France

Oleg Gusikhin
RIC Building
Ford Research and Advanced Engineering
Dearborn, MI
USA

Alain Bernard
Ecole Centrale de Nantes
Nantes Cédex 3
France

Kurosh Madani
Domaine Chérioux
LISSI
Vitry-sur-Seine
France

ISSN 1876-1100

ISBN 978-3-319-03499-7

DOI 10.1007/978-3-319-03500-0

Springer Cham Heidelberg New York Dordrecht London

ISSN 1876-1119 (electronic)

ISBN 978-3-319-03500-0 (eBook)

Library of Congress Control Number: 2013956806

© Springer International Publishing Switzerland 2014

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed. Exempted from this legal reservation are brief excerpts in connection with reviews or scholarly analysis or material supplied specifically for the purpose of being entered and executed on a computer system, for exclusive use by the purchaser of the work. Duplication of this publication or parts thereof is permitted only under the provisions of the Copyright Law of the Publisher's location, in its current version, and permission for use must always be obtained from Springer. Permissions for use may be obtained through RightsLink at the Copyright Clearance Center. Violations are liable to prosecution under the respective Copyright Law. The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

While the advice and information in this book are believed to be true and accurate at the date of publication, neither the authors nor the editors nor the publisher can accept any legal responsibility for any errors or omissions that may be made. The publisher makes no warranty, express or implied, with respect to the material contained herein.

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Preface

The present book includes extended and revised versions of a set of selected papers from the 9th International Conference on Informatics in Control Automation and Robotics (ICINCO 2012), held in Rome, Italy, from 28 to 31 July 2012. The conference was organized in four simultaneous tracks: Intelligent Control Systems and Optimization, Robotics and Automation, Systems Modelling, Signal Processing and Control and Industrial Engineering, Production and Management.

ICINCO 2012 received 360 paper submissions from 58 countries in all continents. From these, after a blind review process, only 40 were accepted as full papers, of which 20 were selected for inclusion in this book, based on the classifications provided by the Program Committee. The selected papers reflect the interdisciplinary nature of the conference as well as the logic equilibrium between the four above-mentioned tracks. The diversity of topics is an important feature of this conference, enabling an overall perception of several important scientific and technological trends. These high-quality standards will be maintained and reinforced at ICINCO 2013, to be held in Reykjavík, Iceland, and in future editions of this conference.

Furthermore, ICINCO 2012 included four plenary keynote lectures given by Alessandro De Luca (Università di Roma “La Sapienza”, Italy), Munther A. Dahleh (MIT, United States), Alexandre Dolgui (Ecole des Mines de Saint-Etienne, France) and Jurek Sasiadek (Carleton University, Canada). We would like to express our appreciation to all of them and in particular to those who took the time to contribute with a paper to this book.

On behalf of the conference organizing committee, we would like to thank all participants. First of all to the authors, whose quality work is the essence of the conference and to the members of the Program Committee, who helped us with their expertise and diligence in reviewing the papers. As we all know, producing a conference requires the effort of many individuals. We wish to thank also all the members of our organizing committee, whose work and commitment were invaluable.

March 2012

Jean-Louis Ferrier
Alain Bernard
Oleg Gusikhin
Kurosh Madani

Organization

Conference Chair

Jean-Louis Ferrier, University of Angers, France

Program Co-chairs

Alain Bernard, Ecole Centrale de Nantes, France

Oleg Gusikhin, Ford Research and Advanced Engineering, USA

Kurosh Madani, University of PARIS-EST Créteil (UPEC), France

Organizing Committee

Marina Carvalho, INSTICC, Portugal

Helder Coelhas, INSTICC, Portugal

Andreia Costa, INSTICC, Portugal

Bruno Encarnação, INSTICC, Portugal

Carla Mota, INSTICC, Portugal

Raquel Pedrosa, INSTICC, Portugal

Vitor Pedrosa, INSTICC, Portugal

Cláudia Pinto, INSTICC, Portugal

Susana Ribeiro, INSTICC, Portugal

José Varela, INSTICC, Portugal

Pedro Varela, INSTICC, Portugal

Program Committee

Arvin Agah, USA

El-Houssaine Aghezzaf, Belgium

Arturo Hernandez Aguirre, Mexico

Eugenio Aguirre, Spain
Hyo-Sung Ahn, Republic of Korea
Adel Al-Jumaily, Australia
Frank Allgower, Germany
Yacine Amirat, France
Stefan Andrei, USA
Peter Arato, Hungary
Rui Araujo, Portugal
Helder Araújo, Portugal
Carles Batlle Arnau, Spain
Tomas Arredondo, Chile
Ronald Askin, USA
T. Asokan, India
Wudhichai Assawinchaichote, Thailand
Ali Bab-Hadiashar, Australia
Jacky Baltes, Canada
Ruth Bars, Hungary
Laxmidhar Behera, India
Karsten Berns, Germany
Arijit Bhattacharya, Ireland
Robert Bicker, UK
Mauro Birattari, Belgium
Christian Blum, Spain
Jean-louis Boimond, France
Magnus Boman, Sweden
Alan Bowling, USA
Thomas Braunl, Australia
Mietek Brdys, UK
Kevin Burn, UK
Amaury Caballero, USA
Javier Fernandez de Canete, Spain
Giuseppe Carbone, Italy
Alessandro Casavola, Italy
Riccardo Cassinis, Italy
Ratchatin Chanchareon, Thailand
Siam Charoenseang, Thailand
Antonio Chella, Italy
Peter C. Y. Chen, Singapore
Wen-Hua Chen, UK
Yuh-Min Chen, Taiwan
Albert Cheng, USA
Graziano Chesi, China
Tsung-Che Chiang, Taiwan
Sung-Bae Cho, Republic of Korea
Ryszard S. Choras, Poland

Chengbin Chu, France
Carlos Coello Coello, Mexico
James M. Conrad, USA
Yechiel Crispin, USA
José Boaventura Cunha, Portugal
Prithviraj (Raj) Dasgupta, USA
Michael A. Demetriou, USA
Mingcong Deng, Japan
Rüdiger Dillmann, Germany
Dimos Dimarogonas, Sweden
Tony Dodd, UK
Alexandre Dolgui, France
António Dourado, Portugal
Venky Dubey, UK
Marc Ebner, Germany
Petr Ekel, Brazil
Mohammed El-Abd, Kuwait
Atilla Elci, Turkey
Simon G. Fabri, Malta
Paolo Falcone, Sweden
David Fernández-Llorca, Spain
Paolo Fiorini, Italy
Juan J. Flores, Mexico
Heinz Frank, Germany
Georg Frey, Germany
Ponnambalam S. G., Malaysia
Dragan Gamberger, Croatia
Nicholas Gans, USA
Maria I. Garcia-Planas, Spain
Ryszard Gessing, Poland
Lazea Gheorghe, Romania
Paulo Gil, Portugal
Maria Gini, USA
Alessandro Giua, Italy
Luis Gomes, Portugal
Bhaskaran Gopalakrishnan, USA
Frans C. A Groen, The Netherlands
Jason Gu, Canada
Kevin Guelton, France
José J. Guerrero, Spain
Eugenio Guglielmelli, Italy
Lei Guo, China
Thomas Gustafsson, Sweden
Maki K. Habib, Egypt
Wolfgang Halang, Germany

John Hallam, Denmark
Jennifer Harding, UK
John Harris, USA
Inman Harvey, UK
Dominik Henrich, Germany
Suranga Hettiarachchi, USA
Victor Hinoestroza, Mexico
Kaoru Hirota, Japan
Wladyslaw Homenda, Poland
Fumiya Iida, Switzerland
Mirjana Ivanovic, Serbia
Sarangapani Jagannathan, USA
Masoud Jamei, UK
Ray Jarvis, Australia
Thira Jearsiripongkul, Thailand
Myong K. Jeong, USA
Ping Jiang, UK
Ivan Kalaykov, Sweden
Michail Kalogiannakis, Greece
Mansour Karkoub, Qatar
Dusko Katic, Serbia
Behrokh Khoshnevis, USA
DaeEun Kim, Republic of Korea
Jonghwa Kim, Germany
Diego Klabjan, USA
Ashok K. Kochhar, UK
Linxue Kong, Australia
Waree Kongprawechnon, Thailand
Israel Koren, USA
George L. Kovács, Hungary
Krzysztof Kozłowski, Poland
Mianowski Krzysztof, Poland
Masao Kubo, Japan
Kin Keung Lai, Hong Kong
Kathryn J. De Laurentis, USA
Soo-Young Lee, Republic of Korea
Graham Leedham, Australia
Kauko Leiviskä, Finland
Kang Li, UK
Tsai-Yen Li, Taiwan
Yangmin Li, China
Gordon Lightbody, Ireland
Huei-Yung Lin, Taiwan
Józef Lisowski, Poland
Changchun Liu, USA

Guoping Liu, UK
Honghai Liu, UK
Savvas Loizou, Cyprus
Luís Seabra Lopes, Portugal
Gonzalo Lopez-Nicolas, Spain
Iuan-Yuan Lu, Taiwan
Edwin Lughofer, Austria
Martin Lukac, Japan
José Tenreiro Machado, Portugal
Anthony Maciejewski, USA
Nitaigour Mahalik, USA
Bruno Maione, Italy
Frederic Maire, Australia
Om Malik, Canada
Jacek Mandziuk, Poland
Philippe Martinet, France
Sonia Martínez, USA
Aníbal Matos, Portugal
Ross McAree, Australia
Ken McGarry, UK
Carlo Menon, Canada
Patrick Millot, France
Sara Moein, Malaysia
António Paulo Moreira, Portugal
Vladimir Mostyn, Czech Republic
Zoltan Nagy, UK
Saeid Nahavandi, Australia
Andreas Nearchou, Greece
Luciana Nedel, Brazil
Sergiu Nedeveschi, Romania
Maria Neves, Portugal
Juan A. Nolasco-Flores, Mexico
Andrzej Obuchowicz, Poland
José Valente de Oliveira, Portugal
Christos Panayiotou, Cyprus
Igor Paromtchik, France
Bozenna Pasik-Duncan, USA
Pierre Payeur, Canada
Fernando Lobo Pereira, Portugal
Marco Antonio Arteaga Perez, Mexico
Jeff Pieper, Canada
Selwyn Piramuthu, USA
Angel P. Del Pobil, Spain
Marie-Noëlle Pons, France
Raul Marin Prades, Spain

Robert Richardson, UK
Julio Elias Normey Rico, Brazil
Juha Rönning, Finland
Mehdi Roopaei, USA
Danilo De Rossi, Italy
Carlos Sagüés, Spain
Mehmet Sahinkaya, UK
Antonio Sala, Spain
Abdel-Badeeh Mohamed Salem, Egypt
Marcello Sanguineti, Italy
Elena De Santis, Italy
Srikanth Saripalli, USA
Medha Sarkar, USA
Nilanjan Sarkar, USA
Jurek Sasiadek, Canada
Hideyuki Sawada, Japan
Daniel Sbarbaro, Chile
Klaus Schilling, Germany
Carla Seatzu, Italy
Ali Selamat, Malaysia
João Sequeira, Portugal
Michael Short, UK
Dan Simon, USA
Olivier Simonin, France
Joaquin Sitte, New Zealand
Adam Slowik, Poland
Andrzej Sluzek, UAE
Qing Song, Singapore
Stefano Squartini, Italy
Burkhard Stadlmann, Austria
Sergiu Stan, Romania
Tarasiewicz Stanislaw, Canada
A. Frank Van Der Stappen, The Netherlands
J. D. Stigter, The Netherlands
Olaf Stursberg, Germany
Kazuya Takeda, Japan
Tianhao Tang, China
József K. Tar, Hungary
Abdelhamid Tayebi, Canada
Daniel Thalmann, Singapore
Haydn A. Thompson, UK
Gui Yun Tian, UK
N. G. Tsagarakis, Italy
Avgoustos Tsinakos, Greece
Antonios Tsourdos, UK

Ali Emre Turgut, Belgium
Angel Valera, Spain
Bram Vanderborght, Belgium
Eloisa Vargiu, Spain
Ramiro Velazquez, Mexico
Damir Vrancic, Slovenia
Bernardo Wagner, Germany
James Whidborne, UK
Sangchul Won, Republic of Korea
Peter Xu, New Zealand
Xun W. Xu, New Zealand
Marek Zaremba, Canada
Janan Zaytoon, France
Du Zhang, USA
Primo Zingaretti, Italy
Loredana Zollo, Italy
Argyrios Zolotas, UK

Auxiliary Reviewers

Francesca Cordella, Italy
Ayan Dutta, USA
Daniel Gutierrez Gomez, Spain
K. R. Guruprasad, USA
Chin-Tien Huang, Taiwan
Janyl Jumadinova, USA
Aravindhan Krishnan, USA
Paolo Lino, Italy
Angelica Muñoz-Meléndez, Mexico
Gulherme Raffo, Brazil
Daniele Rosa, Italy
Jinglin Shen, USA
Merhan Zareh, Italy

Invited Speakers

Alessandro De Luca, Università di Roma “La Sapienza”, Italy
Munther A. Dahleh, MIT, USA
Alexandre Dolgui, Ecole des Mines de Saint-Etienne, France
Jurek Sasiadek, Carleton University, Canada

Contents

Part I Intelligent Control Systems and Optimization

1	Adaptive Flux Observers and Rotor Speed Sensor Fault Detection in Induction Motors	3
	R. Marino, S. Scalzi, P. Tomei and C. M. Verrelli	
2	On Visual Analytics in Plant Monitoring	19
	Tim Tack, Alexander Maier and Oliver Niggemann	
3	Global Optimization for 2D SLAM Problem	35
	Usman Qayyum and Jonghyuk Kim	
4	Stochastic Models and Optimization Algorithms for Decision Support in Spacecraft Control Systems Preliminary Design	51
	Eugene Semenkin and Maria Semenkina	
5	A Heuristic Control Algorithm for Robust Internal Model Control with Arbitrary Reference Model	67
	M. G. Skarpetis, F. N. Koumboulis and A. S. Ntellis	
6	A Multi-Signal Variant for the GPU-Based Parallelization of Growing Self-Organizing Networks	83
	Giacomo Parigi, Angelo Stramieri, Danilo Pau and Marco Piastra	

Part II Robotics and Automation

7	Office Delivery Robot Controlled by Modular Behavior Selection Networks with Planning Capability	103
	Young-Seol Lee and Sung-Bae Cho	
8	Worst-Case Performance Analysis in ℓ_1-norm for an Automated Heavy Vehicle Platoon	115
	Gábor Rödönyi, Péter Gáspár, József Bokor and László Palkovics	

9 Visual SLAM Based on Single Omnidirectional Views 131
David Valiente, Arturo Gil, Lorenzo Fernández and Óscar Reinoso

10 Metrics for Path Planning of Reconfigurable Robots in Uneven Terrain 147
Michael Brunner, Bernd Brüggemann and Dirk Schulz

11 A Combined Direct and Indirect Adaptive Control Scheme for a Wheeled Mobile Robot using Multiple Models 167
Altan Onat and Metin Ozkan

12 Real-Time Visual Servoing Based on New Global Visual Features 183
Laroussi Hammouda, Khaled Kaaniche, Hassen Mekki and Mohamed Chtourou

13 Compliance Error Compensation in Robotic-Based Milling 197
Alexandr Klimchik, Dmitry Bondarenko, Anatol Pashkevich, Sébastien Briot and Benoît Furet

14 A Modified LGMD Based Neural Network for Automatic Collision Detection 217
Ana Carolina Silva, Jorge Silva and Cristina Peixoto dos Santos

15 Vision Based Motion Estimation of Obstacles in Dynamic Unstructured Environments 235
Andrei Vatavu and Sergiu Nedevschi

16 Real-Time Vision-Based Pedestrian Detection in a Truck’s Blind Spot Zone Using a Warping Window Approach 251
Kristof Van Beeck, Toon Goedemé and Tinne Tuytelaars

17 A Proposal of Risk Indexes at Signalised Intersections for ADAS Aimed to Road Safety 265
Bruno Dalla Chiara, Francesco Paolo Deflorio and Serena Cuzzola

Part III Signal Processing, Sensors, Systems Modeling and Control

18 A Component-Oriented Model for Wastewater Pumping Plants 281
Mohamed Abdelati, Felix Felgner and Georg Frey

**19 A System Identification Framework for Modeling Complex
Combustion Dynamics Using Support Vector Machines 297**
 Vijay Manikandan Janakiraman, XuanLong Nguyen,
 Jeff Sterniak and Dennis Assanis

Author Index 315

Part I
Intelligent Control Systems
and Optimization

Chapter 1

Adaptive Flux Observers and Rotor Speed Sensor Fault Detection in Induction Motors

R. Marino, S. Scalzi, P. Tomei and C. M. Verrelli

Abstract The problem of detecting a rotor speed sensor fault in induction motor applications with load torque and rotor/stator resistances uncertainties is addressed. It is shown that in typical operating conditions involving constant rotor speed and flux modulus and non-zero load torque, a constant non-zero (sufficiently large) difference between the measured speed and the actual speed may be on-line identified by an adaptive flux observer which incorporates a convergent rotor resistance identifier and relies on the measured rotor speed and stator currents/voltages. Simulation and experimental results illustrate the effectiveness of the proposed solution and show satisfactory fault detection performances.

Keywords Induction motors · Speed sensor fault · Fault detection · Adaptive observer

1.1 Introduction

The idea underlying a model-based approach to fault diagnosis (see for instance [1, 2]) relies on the assumption that certain process signals carry information about the faults of interest. The gist of the approach is to generate, on the basis

R. Marino · S. Scalzi · P. Tomei · C. M. Verrelli (✉)
Electronic Engineering Department of Tor Vergata University,
Via del Politecnico 1, 00133 Rome, Italy
e-mail: verrelli@ing.uniroma2.it

R. Marino
e-mail: marino@ing.uniroma2.it

S. Scalzi
e-mail: scalzi@ing.uniroma2.it

P. Tomei
e-mail: tomei@ing.uniroma2.it

of measurements from (and knowledge of) the system, a set of “residual signals” which are zero when no fault is present and non-zero when faults occur [3]. However several difficulties naturally arise for the specific application to induction motors: induction motor dynamics are nonlinear; flux measurements are not available; three critical parameters, namely rotor and stator resistances (which vary during operations due to motor heating) and load torque (which depends on applications), are typically uncertain and are required to be on-line estimated. A first intuitive solution \mathcal{IS} to rotor speed sensor fault detection problems relies on designing an adaptive flux/speed observer which does not use the measured speed but only stator currents and voltages measurements (see [4, 5]). The gist of the above design is to compare the measured speed with the estimated one with the aim of identifying the possibly occurring rotor speed sensor fault. Since suitable identifiers for the uncertain parameters (in particular rotor resistance) are to be incorporated in the adaptive observer in order to avoid false fault detections, the drawback of the above approach is then constituted by the well-known identifiability and observability issues which arise when only stator currents and voltages are measured. It is in fact well-established that when the motor typically operates at constant rotor speed and flux modulus with non-zero load torque to minimize power losses and maximize power efficiency at steady-state (see [6]), the simultaneous estimation of rotor speed and rotor resistance cannot be achieved (see [7, 8] as well as [9] and references therein) since only a linear combination $\mathcal{L} = R_r + \gamma\omega$ of the rotor resistance R_r and speed ω (along with the real γ) can be on-line identified by stator currents and voltages measurements. This structural difficulty may be used to our advantage—and this is the novelty of this chapter—by noting that when the (constant) measured speed ω_m is used by a suitable adaptive flux observer \mathcal{AFO} which provides an exponentially convergent rotor resistance estimate when $\omega_m \equiv \omega$, the identifiable linear combination becomes $\mathcal{L}_e = R_r + \gamma(\omega - \omega_m)$. If the rotor speed is measured and no rotor speed sensor fault occurs, i.e. $\omega \equiv \omega_m$, estimating \mathcal{L}_e coincides with estimating R_r ; on the other hand, in the presence of speed sensor failures, estimating \mathcal{L}_e coincides with estimating a quantity which, depending on $(\omega - \omega_m)$, may be larger or smaller than any admissible $R_r \in [R_{rM}, R_{rM}]$ for the specific motor in consideration, that is $R_r + \gamma(\omega - \omega_m) < R_{rM}$ or $R_r + \gamma(\omega - \omega_m) > R_{rM}$. In this case a rotor speed sensor fault may be on-line identified by designing a speed measurement-based adaptive observer and by monitoring the estimate of \mathcal{L}_e on the basis of the boundary values R_{rM} and R_{rM} .

The contribution of this chapter is then to show that an adaptive flux observer \mathcal{AFO} which incorporates a convergent rotor resistance identifier and relies on the measured rotor speed and stator currents/voltages may be effectively used to on-line identify a constant non-zero (sufficiently large) difference $\omega - \omega_m$: (i) in typical operating conditions involving non-zero load torque and constant rotor speed and flux modulus; (ii) in the presence of load torque and stator resistance uncertainties. In particular, denoting by $\alpha \in [\alpha_m, \alpha_M]$ ($\alpha_m = L_r^{-1} R_{rM}$, $\alpha_M = L_r^{-1} R_{rM}$) the ratio between the rotor resistance R_r and the rotor inductance L_r and by $\hat{\alpha}$ its estimate (provided by the \mathcal{AFO}), we will show that a residual signal for the rotor speed sensor fault detection may be chosen as the steady-state distance of $\hat{\alpha}(t)$ from the

compact set $[\alpha_m, \alpha_M]$, i.e. $\lim_{t \rightarrow +\infty} \text{dist}(\hat{\alpha}(t); [\alpha_m, \alpha_M])$. A similar idea (though not analytically motivated) has been recently presented in [10] even though it relies on an observer which is only adaptive with respect to the rotor resistance. In contrast to [10], we propose a candidate adaptive flux observer belonging to the set of all adaptive flux observers which provide convergent estimates of the rotor resistance despite uncertainties in critical parameters such as load torque and stator resistance. This is to avoid false fault detections which may be related to uncertainties in those critical parameters. With this respect, among the adaptive observers which have been proposed in the literature since 1978 (see for instance [10–17] and references therein), we consider in this chapter the one presented in [16] which is simultaneously characterized by: (i) an overall structural simplicity with no use of sign functions, high gains or output time derivatives which lead to well-known implementation difficulties and high noise sensitivity; (ii) persistency of excitation conditions which are naturally related to motor observability and parameter identifiability and are guaranteed to be satisfied in the typical case of constant motor speed and flux modulus and non-zero electro-magnetic torque; (iii) exponential convergence properties guaranteeing a certain degree of robustness. It is constituted by an adaptive flux observer which is able to estimate the motor fluxes and to identify the rotor resistance and by a stator resistance identifier whose design is performed on a different time scale in order: (i) to isolate its estimation from the estimation of motor fluxes and rotor resistance (see also [18, 19] for a similar approach to parameter estimation in induction motors); (ii) to not destroy the identifiability property owned by the linear combination \mathcal{L}_e . Simulation and experimental results illustrate the effectiveness of the proposed solution and show satisfactory fault detection performances.

1.2 Physical Modeling

Assuming linear magnetic circuits, the dynamics of a balanced non-saturated induction motor with one pole pair in a fixed reference frame attached to the stator are given by the well known fifth-order model (see for instance [9] and references therein)

$$\begin{aligned}
 \frac{d\omega}{dt} &= \mu(\psi_{ra}i_{sb} - \psi_{rb}i_{sa}) - \frac{T_L}{J} \\
 \frac{d\psi_{ra}}{dt} &= -\alpha\psi_{ra} - \omega\psi_{rb} + \alpha M i_{sa} \\
 \frac{d\psi_{rb}}{dt} &= -\alpha\psi_{rb} + \omega\psi_{ra} + \alpha M i_{sb} \\
 \frac{di_{sa}}{dt} &= -\left(\frac{R_s}{\sigma} + \beta\alpha M\right)i_{sa} + \frac{1}{\sigma}u_{sa} + \beta\alpha\psi_{ra} + \beta\omega\psi_{rb} \\
 \frac{di_{sb}}{dt} &= -\left(\frac{R_s}{\sigma} + \beta\alpha M\right)i_{sb} + \frac{1}{\sigma}u_{sb} + \beta\alpha\psi_{rb} - \beta\omega\psi_{ra}
 \end{aligned} \tag{1.1}$$

in which: ω is the rotor speed, (ψ_{ra}, ψ_{rb}) are the rotor fluxes, (i_{sa}, i_{sb}) are the stator currents, (u_{sa}, u_{sb}) are the stator voltages in a fixed reference attached to the stator. The constant model parameters are: load torque T_L ; motor moment of inertia J ; rotor and stator windings resistances (R_r, R_s) and inductances (L_r, L_s) ; mutual inductance M . To simplify notations we use the reparameterization: $\alpha = \frac{R_r}{L_r}$, $\beta = \frac{M}{\sigma L_r}$, $\mu = \frac{M}{J L_r}$, $\sigma = L_s(1 - \frac{M^2}{L_s L_r})$. The rotor fluxes (ψ_{ra}, ψ_{rb}) are unmeasured variables since flux sensors are usually not available while the parameters T_L , α and R_s are typically uncertain owing to load torque dependence on applications and owing to resistance variations which depend on motor heating. We will only assume, in the following, the boundedness of state and input variables while any restriction concerning the boundedness of stator currents integrals, which has been proposed for the design of similar adaptive flux observers in [13, 15], is not required.

1.3 Observer Design

The first idea in [16] is to introduce the variables $z_a = i_{sa} + \beta\psi_{ra}$, $z_b = i_{sb} + \beta\psi_{rb}$ so that the motor electro-magnetic equations in (1.1) become

$$\begin{aligned} \dot{z}_a &= -\frac{R_s}{\sigma}i_{sa} + \frac{1}{\sigma}u_{sa} \\ \dot{z}_b &= -\frac{R_s}{\sigma}i_{sb} + \frac{1}{\sigma}u_{sb} \\ \frac{di_{sa}}{dt} &= -\frac{R_s}{\sigma}i_{sa} - \alpha(1 + \beta M)i_{sa} - \omega i_{sb} + \alpha z_a + \omega z_b + \frac{1}{\sigma}u_{sa} \\ \frac{di_{sb}}{dt} &= -\frac{R_s}{\sigma}i_{sb} - \alpha(1 + \beta M)i_{sb} + \omega i_{sa} + \alpha z_b - \omega z_a + \frac{1}{\sigma}u_{sb}. \end{aligned} \quad (1.2)$$

The advantage of using the (z_a, z_b) variables, which are physically related to the stator fluxes, is that their dynamics depend neither on the unmeasured rotor fluxes nor on the uncertain rotor resistance. On the basis of model (1.2), the following observer is designed (k_i is a positive design parameter):

$$\begin{aligned} \dot{\hat{i}}_{sa} &= -\frac{\hat{R}_s}{\sigma}i_{sa} - \hat{\alpha}(1 + \beta M)i_{sa} - \omega \hat{i}_{sb} + \hat{\alpha} \hat{z}_a + \omega \hat{z}_b + \frac{u_{sa}}{\sigma} + k_i(i_{sa} - \hat{i}_{sa}) \\ \dot{\hat{i}}_{sb} &= -\frac{\hat{R}_s}{\sigma}i_{sb} - \hat{\alpha}(1 + \beta M)i_{sb} + \omega \hat{i}_{sa} + \hat{\alpha} \hat{z}_b - \omega \hat{z}_a + \frac{u_{sb}}{\sigma} + k_i(i_{sb} - \hat{i}_{sb}) \\ \dot{\hat{z}}_a &= -\frac{\hat{R}_s}{\sigma}i_{sa} + \frac{u_{sa}}{\sigma} + v_a \\ \dot{\hat{z}}_b &= -\frac{\hat{R}_s}{\sigma}i_{sb} + \frac{u_{sb}}{\sigma} + v_b \end{aligned} \quad (1.3)$$

which is a copy of system (1.2)¹ with: (i) the estimates $(\hat{z}_a, \hat{z}_b, \hat{\alpha}, \hat{R}_s)$ in place of the unmeasured/uncertain (z_a, z_b, α, R_s) ; (ii) stabilizing terms on the current estimation errors $(i_{sa} - \hat{i}_{sa}), (i_{sb} - \hat{i}_{sb})$; (iii) the compensating terms v_a, v_b yet to be designed. According to the stability analysis in [16], the estimation laws for $\hat{\alpha}$ and \hat{R}_s and the feedback terms v_a, v_b are chosen as

$$\begin{aligned}\dot{\hat{\alpha}} &= -k_\alpha \left[[(1 + \beta M)i_{sa} - \hat{z}_a] \tilde{i}_{sa} + [(1 + \beta M)i_{sb} - \hat{z}_b] \tilde{i}_{sb} \right] \\ \dot{\hat{R}}_s &= -k_R (v_a i_{sa} + v_b i_{sb}) \\ v_a &= -k_z (\omega \tilde{i}_{sb} - \hat{\alpha} \tilde{i}_{sa}) \\ v_b &= k_z (\omega \tilde{i}_{sa} + \hat{\alpha} \tilde{i}_{sb})\end{aligned}\quad (1.4)$$

in which $\tilde{i}_{sa} = i_{sa} - \hat{i}_{sa}$, $\tilde{i}_{sb} = i_{sb} - \hat{i}_{sb}$ are the stator current estimation errors, k_z and k_α are positive design parameters, k_R is a sufficiently small positive design parameter.

Remark 1 While the design parameter k_i directly affects the dynamics of the stator current estimation errors, k_α, k_R are the adaptation gains for the estimates $\hat{\alpha}, \hat{R}_s$ while k_z characterizes the influence of stator current estimation errors on the dynamics of the (z_a, z_b) -estimation errors.

The two-time-scale arguments in [16], under certain identifiability assumptions at steady-state, allow for isolating the estimation of the stator resistance from the estimation of motor fluxes (achieved through the estimation of (z_a, z_b)) and rotor resistance so that the following persistency of excitation condition:

\mathcal{P}_e : there exist two positive reals t_p and K_p such that the persistency of excitation condition (I_3 is the 3×3 identity matrix)

$$\int_t^{t+t_p} \Gamma^T(\tau) \Gamma(\tau) d\tau \geq K_p I_3, \quad \forall t \geq 0 \quad (1.5)$$

holds with

$$\Gamma = \begin{bmatrix} \alpha & \omega & \beta (\psi_{ra} - M i_{sa}) \\ -\omega & \alpha & \beta (\psi_{rb} - M i_{sb}) \end{bmatrix}$$

is obtained. Inequality (1.5) is naturally related to motor observability and parameter identifiability: when the rotor speed and the rotor flux modulus are constant and the

¹ The terms $-\omega \hat{i}_{sb}$ and $\omega \hat{i}_{sa}$ in the first two equations of (1.3) compensate for the rotor back electro-motive forces with the estimates $(\hat{i}_{sa}, \hat{i}_{sb})$ in place of (i_{sa}, i_{sb}) leading to skew-symmetric terms in the estimation error dynamics.

load torque is zero so that $\psi_{ra} = Mi_{sa}$ and $\psi_{rb} = Mi_{sb}$, it cannot be satisfied²; when a positive load torque is applied³ and when the rotor speed and the rotor flux modulus are constant⁴ with $\psi_{ra}^2 + \psi_{rb}^2 = c_\psi > 0$, it is satisfied.

The gist of the estimation design in [16] and the required assumptions can be simply explained in the following terms: if the adaptive observer (1.3)–(1.4) (with no stator resistance identifier) is used with a constant value of the stator resistance that is slightly different from its actual value, then a non-zero steady-state solution appears that causes a suitable measured output function $s_\pi = v_a i_{sa} + v_b i_{sb}$ to be, in first approximation, monotone with respect to the R_s -estimation error $\tilde{R}_s = R_s - \hat{R}_s$; thus, by adjusting the R_s -estimate \hat{R}_s on the basis of this output function (slowly, in order not to deviate too much from the steady-state solution) one can obtain the correct estimation of R_s and the consequent exponential convergence to zero of all the estimation errors $\tilde{i}_{sa}, \tilde{i}_{sb}, z_a - \hat{z}_a, z_b - \hat{z}_b, \alpha - \hat{\alpha}, \tilde{R}_s$. Exponential rotor flux recovering can be finally obtained by

$$\begin{bmatrix} \hat{\psi}_{ra} \\ \hat{\psi}_{rb} \end{bmatrix} = -\frac{1}{\beta} \begin{bmatrix} \hat{i}_{sa} - \hat{z}_a \\ \hat{i}_{sb} - \hat{z}_b \end{bmatrix}$$

where the filtered estimates $(\hat{i}_{sa}, \hat{i}_{sb})$ are preferred to the measured (i_{sa}, i_{sb}) for practical implementation issues. The following second-order load torque identifier ($k_{\omega e}$ and k_T are positive design parameters):

$$\begin{aligned} \dot{\hat{\omega}} &= \mu(\hat{\psi}_{ra} i_{sb} - \hat{\psi}_{rb} i_{sa}) - \frac{\hat{T}_L}{J} + k_{\omega e}(\omega - \hat{\omega}) \\ \dot{\hat{T}}_L &= -k_T(\omega - \hat{\omega}) \end{aligned}$$

is finally proposed in [16]. It can be used in conjunction with the adaptive observer (1.3)–(1.4) to provide an exponentially convergent estimate of the load torque once convergent estimates of rotor fluxes have been obtained. The proof, which is reported in [9], is based on the quadratic function

$$V_T = \frac{1}{2k_T J} \tilde{T}_L^2 + \frac{1}{2} \tilde{\omega}^2 + \epsilon_T \tilde{\omega} \tilde{T}_L$$

in which $\tilde{\omega} = \omega - \hat{\omega}$, $\tilde{T}_L = T_L - \hat{T}_L$ and $\epsilon_T > 0$ is a sufficiently small positive real.

² Recall that in these operating conditions the rotor resistance cannot be identified by stator currents and rotor speed measurements since the motor equations (1.1) become

$$\dot{\omega} = 0, \quad \dot{\psi}_{ra} = -\omega \psi_{rb}, \quad \dot{\psi}_{rb} = \omega \psi_{ra}, \quad \frac{di_{sa}}{dt} = -\omega i_{sb}, \quad \frac{di_{sb}}{dt} = \omega i_{sa}$$

and do not depend on the rotor resistance R_r .

³ A negative load torque (for regenerative brake actions) is also allowed provided that a non-zero rotor flux vector speed results.

⁴ It suffices that at least they asymptotically tend to constant values with time derivatives asymptotically converging to zero.

1.4 Speed Sensor Fault Detection

The aim of this section is to prove that a constant non-zero (sufficiently large) difference between the measured speed and the actual speed may be on-line identified by the adaptive flux observer (1.3)–(1.4) (in which the measured speed ω_m replaces the actual speed ω) in typical operating conditions involving non-zero load torque and constant rotor speed and flux modulus. To this purpose, we preliminarily note that in those conditions we have

$$\begin{aligned} \overbrace{\psi_{ra}^2 + \psi_{rb}^2} &\equiv 0 \\ \dot{\omega} &\equiv 0, \end{aligned}$$

from which we obtain

$$\begin{aligned} Mi_{sa} &= \psi_{ra} - c\psi_{rb} \\ Mi_{sb} &= \psi_{rb} + c\psi_{ra} \end{aligned} \quad (1.6)$$

with (ω_s is the slip speed see [9])⁵

$$c = \frac{T_L M}{J \mu c_\psi} = \frac{T_L L_r}{c_\psi} = \omega_s / \alpha.$$

By adding and subtracting in (1.2) suitable terms proportional to $\omega_e = \omega - \omega_m$ and by using (1.6), Eq. (1.2) can be equivalently rewritten as

$$\begin{aligned} \dot{z}_a &= -\frac{R_s}{\sigma} i_{sa} + \frac{1}{\sigma} u_{sa} \\ \dot{z}_b &= -\frac{R_s}{\sigma} i_{sb} + \frac{1}{\sigma} u_{sb} \\ \frac{di_{sa}}{dt} &= -\frac{R_s}{\sigma} i_{sa} - \omega_m i_{sb} + \omega_m z_b + \frac{1}{\sigma} u_{sa} + \alpha [z_a - (1 + \beta M) i_{sa}] + \omega_e (z_b - i_{sb}) \\ \frac{di_{sb}}{dt} &= -\frac{R_s}{\sigma} i_{sb} + \omega_m i_{sa} - \omega_m z_a + \frac{1}{\sigma} u_{sb} + \alpha [z_b - (1 + \beta M) i_{sb}] - \omega_e (z_a - i_{sa}) \end{aligned}$$

with the last two equations reading

$$\begin{aligned} \frac{di_{sa}}{dt} &= -\frac{R_s}{\sigma} i_{sa} - \omega_m i_{sb} + \omega_m z_b + \frac{1}{\sigma} u_{sa} + \overbrace{\left(\alpha + \frac{\omega_e}{c} \right)}^{\alpha_e} (z_a - (1 + \beta M) i_{sa}) \\ \frac{di_{sb}}{dt} &= -\frac{R_s}{\sigma} i_{sb} + \omega_m i_{sa} - \omega_m z_a + \frac{1}{\sigma} u_{sb} + \overbrace{\left(\alpha + \frac{\omega_e}{c} \right)}^{\alpha_e} (z_b - (1 + \beta M) i_{sb}). \end{aligned}$$

⁵ When no load torque is applied (or equivalently when a zero slip speed results), we have $Mi_{sa} = \psi_{ra}$, $Mi_{sb} = \psi_{rb}$ as preliminarily discussed by footnote 2.

In other terms, in typical operating conditions involving non-zero load torque and constant rotor flux modulus and (measured and actual) speeds, an equivalent constant $\alpha_e = \alpha + \omega_e/c$ appears in the motor model with ω_m in place of ω (compare it with model (1.2)): it incorporates any possibly non-zero difference between the measured speed and the actual speed.

Remark 2 When $\omega_m \equiv 0$ (which is the well-known sensorless scenario in which only stator currents and voltages are measured), α_e reduces to the linear combination $\alpha + \omega/c$ which, as discussed in the Introduction, is the only quantity to be identifiable by stator currents and voltages measurements (see [7–9] and references therein). Any solution to rotor speed sensor fault detection problems relying on adaptive flux/speed observers which do not use the measured speed (but try to estimate it for comparison) thus becomes unfeasible, at least in typical motor operating conditions with rotor resistance uncertainties.

By virtue of the same analysis presented in [16] and discussed in Sect. 1.3 (with ω_m in place of ω), the adaptive observer (1.3)–(1.4) is able to guarantee exponential convergence⁶ to zero of all the estimation errors $\tilde{i}_{sa}, \tilde{i}_{sb}, z_a - \hat{z}_a, z_b - \hat{z}_b, \alpha_e - \hat{\alpha}, \tilde{R}_s$ (provided that initial errors belong to the region of attraction of the origin for the error system dynamics⁷). Exponential estimation of motor fluxes, equivalent α and stator

⁶ Recall from [16] that the proof of convergence is not constrained to the positiveness of the parameter α_e .

⁷ Note that in the considered conditions (non-zero load torque and constant rotor speed and (non-zero) rotor flux modulus) it is possible to only locally identify the uncertain R_r, R_s, T_L from the measured outputs (i_{sa}, i_{sb}, ω) . In fact, according to Sect. 1.3 of [9]:

- R_r, R_s and T_L can be expressed in terms of the measured outputs and their time derivatives ($\dot{i}_{sa,d} = di_{sa}/dt, i_{sb,d} = di_{sb}/dt$) as solutions to the system of nonlinear equations

$$\begin{aligned} \mathcal{P} &= i_{sa}(t_*)\sqrt{u_{sa}^2(t_*) + u_{sb}^2(t_*)} \\ \mathcal{Q} &= i_{sb}(t_*)\sqrt{u_{sa}^2(t_*) + u_{sb}^2(t_*)} \\ \dot{\rho}^* - \frac{R_r T_L}{\psi_r^2} &= \omega \\ \dot{\rho}^* &= \frac{-i_{sa,d}(t_*)i_{sb}(t_*) + i_{sb,d}(t_*)i_{sa}(t_*)}{i_{sa}^2(t_*) + i_{sb}^2(t_*)} \\ \mathcal{V}^2 &= u_{sa}^2(t_*) + u_{sb}^2(t_*) \end{aligned}$$

where: $\mathcal{P} = u_{sd}i_{sd} + u_{sq}i_{sq}$ and $\mathcal{Q} = -u_{sq}i_{sd} + u_{sd}i_{sq}$ are proportional to the active and reactive electrical powers, respectively; $\mathcal{V} = \sqrt{u_{sd}^2 + u_{sq}^2}$ is the modulus of the stator voltage

vector; $\psi_r = \sqrt{\psi_{ra}^2 + \psi_{rb}^2}$ is the modulus of the rotor flux vector; the constant $u_{sd}, u_{sq}, i_{sd}, i_{sq}$ are the (d, q) -components of the stator voltage and current vectors which are known functions of ψ_r, R_r, R_s, T_L (see Sect. 1.3 of [9]); t_* is such that $u_{sa}(t_*) = \mathcal{V}, u_{sb}(t_*) = 0$;

- there may exist two possible solutions $(\psi_r, R_{r1}, R_{s1}, T_{L1}), (\psi_r, R_{r2}, R_{s2}, T_{L2})$ with $R_{r1} = -R_{r2}$ and $T_{L1} = -T_{L2}$ to the above system of nonlinear equations to which correspond the same output and input profiles.

resistance are therefore achieved. In particular, $\lim_{t \rightarrow +\infty} \text{dist}(\hat{\alpha}(t); [\alpha_m, \alpha_M])$ may be used as a residual signal for rotor speed sensor fault detection since when rotor speed ω and flux modulus $\sqrt{\psi_{ra}^2 + \psi_{rb}^2}$ are constant along with the measured speed ω_m , the α -estimate $\hat{\alpha}(t)$ is guaranteed to exponentially converge to α_e . It is clear that only the rotor speed sensor faults that lead to a value of α_e outside the compact set $[\alpha_m, \alpha_M]$ can be in this way identified. An estimate of ω_e (and therefore of the sensor failure magnitude) can be finally obtained according to $\frac{T_L L_r}{c_\psi} (\alpha_e - \alpha) = \omega_e$ once the load torque (through the load torque identifier) and the rotor fluxes have been estimated.⁸

Remark 3 Two relevant consequences of the previous analysis are the following: (i) the rotor fluxes, the load torque (and therefore the motor torque) and the stator resistance can be actually estimated, in the conditions above, even in the presence of rotor speed sensor faults (including the sensorless scenario); (ii) the rotor fluxes, the load torque, the stator resistance and the rotor speed difference $\omega - \omega_m$ (or equivalently the rotor speed ω in the sensorless scenario) can be actually estimated, in the conditions above, provided that the rotor resistance (or equivalently α) is known (see \mathcal{IS} and the related results in [20] for the sensorless scenario).

Remark 4 The key idea of the approach presented in this chapter may be alternatively (equivalently) realized as follows: suppose that no speed information is used by the adaptive flux observer \mathcal{AFO} so that \mathcal{L} along with γ are on-line identified (see the Introduction). Then the information contained in the measured speed ω_m (provided by the sensor) may be successfully employed to compute $\mathcal{L}_e = \mathcal{L} - \gamma\omega_m$. The rotor speed sensor fault can be finally identified as before. This constitutes an improved modification of the previously described intuitive solution \mathcal{IS} .⁹

1.5 Simulation Results

The aim of this section is to illustrate by simulations the previously presented results even in the presence of time-varying perturbations of the motor resistances and step-wise variations of the load torque. To this purpose, the nonlinear adaptive observer (1.3)–(1.4) and the load torque identifier are simulated for the three-phase single pole pair 0.6-kW induction motor OE-MER 7-80/C in [9] whose parameters are: $J = 0.0075 \text{ kgm}^2$, $R_s = 5.3 \text{ Ohm}$, $R_r = 3.3 \text{ Ohm}$, $L_s = 0.365 \text{ H}$, $L_r = 0.375 \text{ H}$, $M = 0.34 \text{ H}$. The motor (with initial conditions $\psi_{ra}(0) = \psi_{rb}(0) = 0.1 \text{ Wb}$) is illustratively controlled by the input-output feedback linearizing control reported in Sect. 2.4 of [9] (which relies on exact rotor speed and stator current measurements

⁸ Note that, for constant ω and ω_m and convergent rotor fluxes estimates, exponential convergence to zero of \tilde{T}_L and of $\omega_m - \hat{\omega}$ can be proved by using the quadratic function V_T with $\omega_m - \hat{\omega}$ in place of $\hat{\omega}$.

⁹ When α is known the two approaches are equivalent.

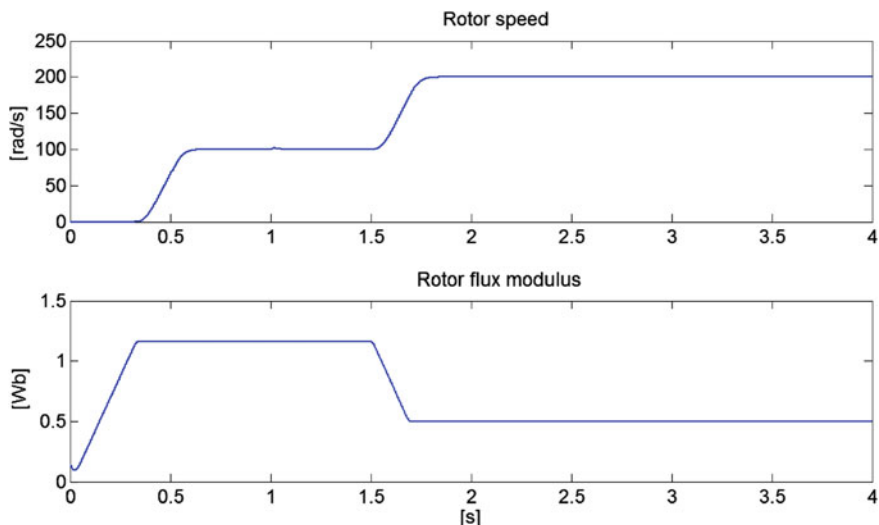


Fig. 1.1 Rotor speed ω and flux modulus $\sqrt{\psi_{ra}^2 + \psi_{rb}^2}$

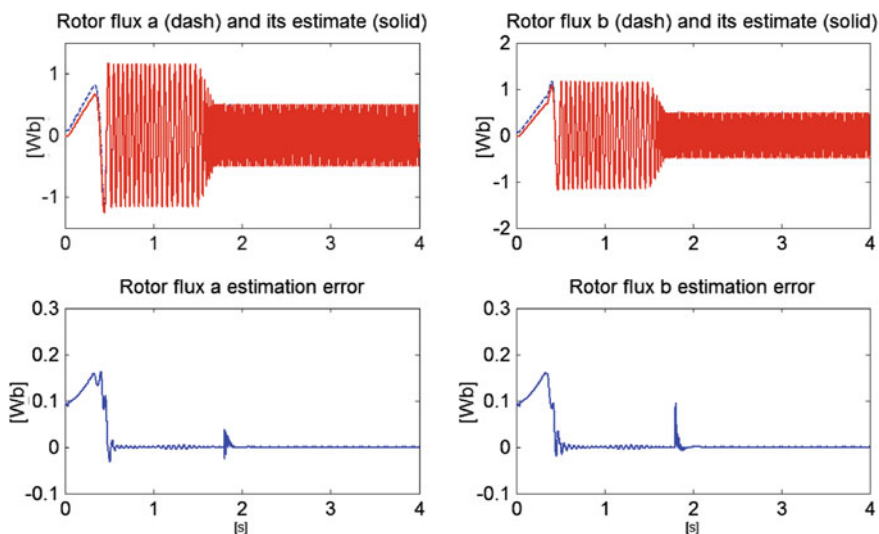


Fig. 1.2 Rotor fluxes ψ_{ra} , ψ_{rb} (dash) and their estimates $\hat{\psi}_{ra}$, $\hat{\psi}_{rb}$ (solid); rotor fluxes ψ_{ra} , ψ_{rb} estimation errors

and on the perfect knowledge of all motor parameters). The rotor speed and the flux modulus are reported in Fig. 1.1.

The design parameters are chosen as (all the values are in SI units): $k_i = 120$, $k_z = 3$, $k_\alpha = 450$, $k_R = 0.1$, $k_\omega = 200$, $k_T = 100^2$ J. All the observer initial

Fig. 1.3 Load torque T_L (dash) and its estimate \hat{T}_L (solid)

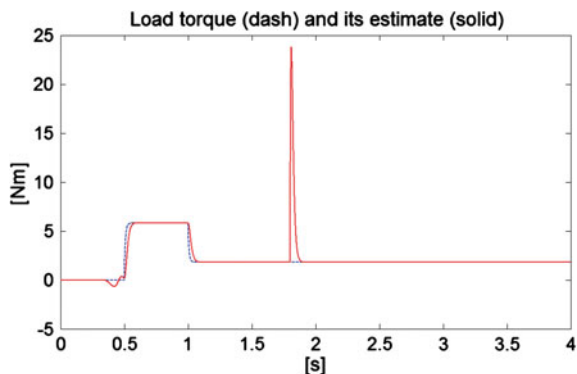
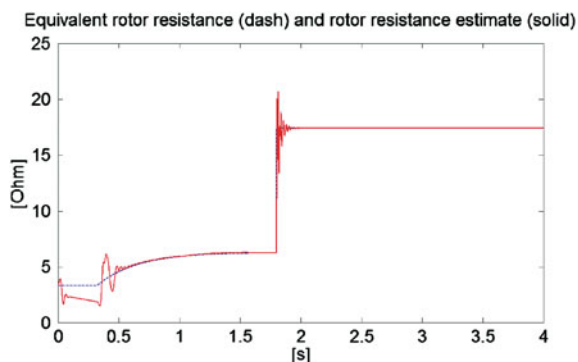


Fig. 1.4 Equivalent rotor resistance R_{re} (dash) and its estimate \hat{R}_r (solid)



conditions are set to zero excepting for $\hat{\alpha}(0) = 9 \text{ s}^{-1}$ and $\hat{R}_s(0) = 5.4 \text{ Ohm}$. For $t < 1.8 \text{ s}$ the measured speed is equal to the actual one ($\omega \equiv \omega_m$) while for $t \geq 1.8 \text{ s}$ a rotor speed sensor fault occurs leading to $\omega - \omega_m = 0.4\omega$. The equivalent rotor resistance $R_e = L_r \alpha_e$ is thus equal to R_r for $t < 1.8 \text{ s}$ and equal to $R_r + \omega_e c_\psi / T_L$ for $t \geq 1.8 \text{ s}$. The rotor fluxes, the load torque, the equivalent rotor resistance, the stator resistance along with the corresponding converging estimates are reported in Figs. 1.2, 1.3, 1.4, 1.5.

Fast estimation is obtained: the rotor speed sensor fault can be promptly identified by monitoring the estimated rotor resistance on the basis of the available bounding values $R_{rM} = L_r \alpha_m = 2.8 \text{ Ohm}$, $R_{rM} = L_r \alpha_M = 6.9 \text{ Ohm}$. In order to illustrate the possibility of detecting false faults by using adaptive observers with no stator resistance identifier (as in [10]), the same simulation is carried out in the presence of no rotor speed sensor fault for the adaptive observer (1.3)–(1.4) with the stator resistance value 5.4 Ohm in place of \hat{R}_s .

As illustrated by Fig. 1.6, a non-zero residual results even in the case of no rotor speed sensor fault¹⁰: this is only due to stator resistance uncertainties and motivates

¹⁰ Even though in this case steady-state stator currents estimation errors may appear (as in [10]), the presence in practice of unavoidable measurements noise which forces those steady-state estimation

Fig. 1.5 Stator resistance R_s (dash) and its estimate \hat{R}_s (solid)

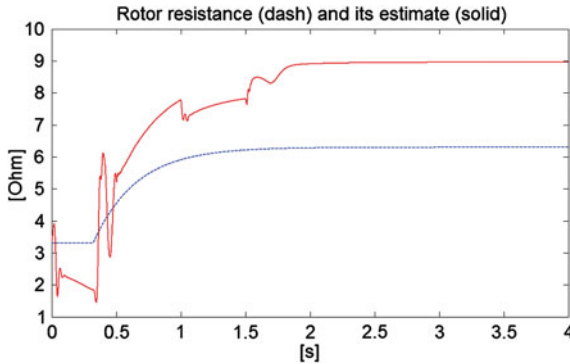
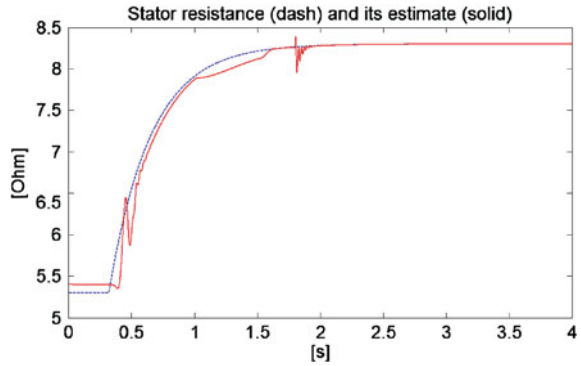


Fig. 1.6 Observer with no R_s -adaptation: rotor resistance R_r (dash) and its estimate \hat{R}_r (solid)

the use of adaptive observers which provide convergent rotor resistance estimates despite stator resistance uncertainties.

1.6 Experimental Results

In this section we present the results of three experimental tests which have been carried out with reference to a 0.25 kW C4T34FB5B Leeson induction motor driven by a 20 kHz PWM-based open loop voltage/frequency control (61 V, 16.7 Hz). The applied load torque, which is proportional to the induction motor speed, is provided by the WSM-3-32-1 Sangalli Servomotori DC permanent magnet motor which is directly connected to the shaft of the induction motor. The nonlinear adaptive observer (1.3)–(1.4) and the load torque identifier are executed (at 12.5 kHz) when the motor has reached its steady-state. The nominal values of the parameters (provided

(Footnote 10 continued)

errors to always be not identically zero makes not reliable the approach of using them as additional residuals.

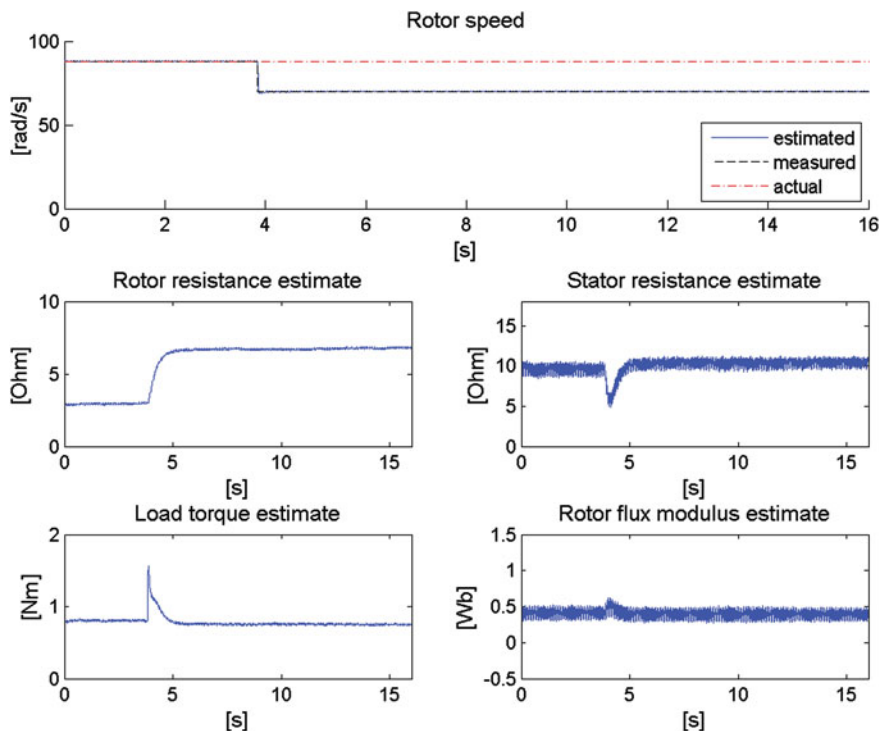


Fig. 1.7 First experimental test ($\omega_m = 70$ rad/s, $\omega = 88$ rad/s)

by the manufacturer) $L_s = 0.268$ H, $L_r = 0.298$ H, $M = 0.258$ H, $J = 0.005$ kgm² are used, the control gains (all the values are in SI units) $k_i = 600$, $k_z = 3$, $k_\alpha = 10$, $k_R = 0.65$, $k_{\hat{\omega}} = 100$, $k_T = 5$ are chosen while zero initial conditions, excepting for $\hat{\alpha}(0)$ and $\hat{R}_s(0)$ (equal to 12.75 s⁻¹ and 10.45 Ohm), are set. The first test, whose steady-state results are reported in Fig. 1.7, involves a partial rotor speed sensor fault ($\omega_m = 70$ rad/s, $\omega = 88$ rad/s) occurring at $t = 3.84$ s.

The second test, whose steady-state results are reported in Fig. 1.8, involves a larger partial rotor speed sensor fault ($\omega_m = 44$ rad/s, $\omega = 88$ rad/s) occurring at $t = 2.02$ s.

The third test, whose steady-state results are reported in Fig. 1.9, finally involves a full rotor speed sensor fault ($\omega_m = 0$ rad/s, $\omega = 88$ rad/s) occurring at $t = 1.66$ s. All the three tests confirm the theoretical results presented in the chapter: as expected, the rotor speed sensor fault can be promptly identified ($R_{rM} = 6$ Ohm) by monitoring the estimated rotor resistance which converges to α_e .¹¹

¹¹ In accordance with the \mathcal{P}_e condition and the related analysis, different transient behaviours result.

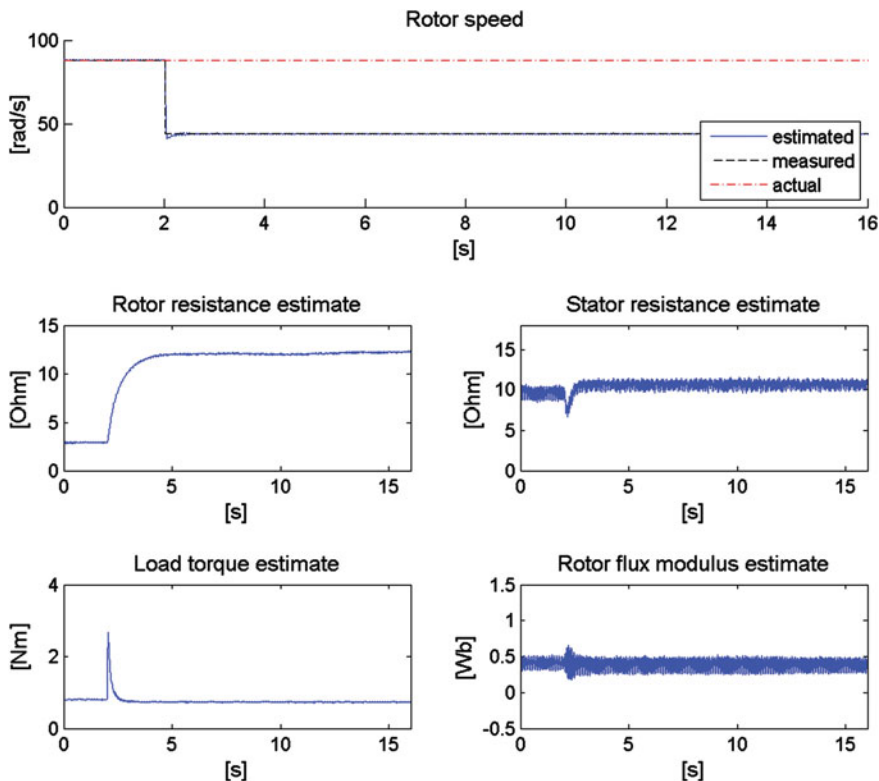


Fig. 1.8 Second experimental test ($\omega_m = 44 \text{ rad/s}$, $\omega = 88 \text{ rad/s}$)

1.7 Conclusions

A constant non-zero (sufficiently large) difference between the measured speed and the actual speed may be on-line identified, even in the presence of uncertainties in load torque and rotor/stator resistances, in typical operating conditions involving non-zero load torque, constant rotor speed and flux modulus. An adaptive flux observer which incorporates a convergent rotor resistance identifier and relies on the measured rotor speed and stator currents/voltages is used to this purpose: it can be incorporated in induction motors control schemes as an effective detector of rotor speed sensor faults. Simulation and experimental results illustrate the effectiveness of the proposed solution and show satisfactory fault detection performances.

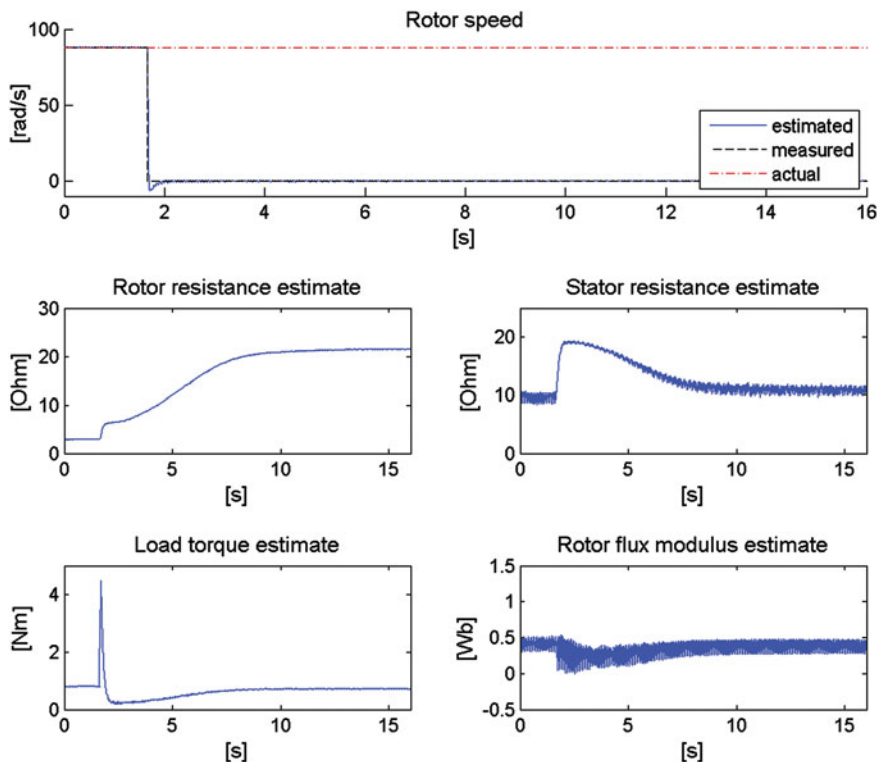


Fig. 1.9 Third experimental test ($\omega_m = 0 \text{ rad/s}$, $\omega = 88 \text{ rad/s}$)

References

1. Isermann, R.: Fault-Diagnosis Applications. Springer, Berlin (2011)
2. Ding, S.X.: Model-based Fault Diagnosis Techniques. Springer, Berlin (2008)
3. Bennet, S.M., Patton, R.J., Daley, S.: Sensor fault-tolerant control of a rail traction drive. *Control Eng. Pract.* **7**, 217–225 (1999)
4. Guzinski, J., Abu-Rub, H., Diguët, M., Krzeminski, Z., Lewicki, A.: Speed and load torque observer application in high-speed train electric drive. *IEEE Trans. Ind. Electron.* **57**, 565–574 (2010)
5. Guzinski, J., Diguët, M., Krzeminski, Z., Lewicki, A., Abu-Rub, H.: Application of speed and load torque observers in high-speed train drive for diagnostic purposes. *IEEE Trans. Ind. Electron.* **56**, 248–256 (2009)
6. Marino, R., Tomei, P., Verrelli, C.M.: Adaptive output feedback tracking control for induction motors with uncertain load torque and resistances. Paper Presented at the International Symposium on Power Electronics, Electrical Drives, Automation and Motion, Pisa, Italy, pp. 419–424 (2010)
7. Ha, I.-J., Lee, S.-H.: An online identification method for both stator and rotor resistances of induction motors without rotational transducers. *IEEE Trans. Ind. Electron.* **47**, 842–853 (2000)
8. Marino, R., Tomei, P., Verrelli, C.M.: An adaptive tracking control from current measurements for induction motors with uncertain load torque and rotor resistance. *Automatica* **44**, 2593–2599 (2008)

9. Marino, R., Tomei, P., Verrelli, C.M.: *Induction Motor Control Design*. Springer, London (2010)
10. Najafabadi, T.A., Salmasi, F.R., Jabejdar-Maralani, P.: Detection and isolation of speed-, DC-link voltage-, and current-sensor faults based on an adaptive observer in induction-motor drives. *IEEE Trans. Ind. Electron.* **58**, 1662–1672 (2011)
11. Castaldi, P., Geri, W., Montanari, M., Tilli, A.: A new adaptive approach for on-line parameter and state estimation of induction motors. *Control Eng. Pract.* **13**, 81–94 (2005)
12. Hasan, S.M.N., Husain, I.: A Luenberger-sliding mode observer for online parameter estimation and adaptation in high-performance induction motor drives. *IEEE Trans. Ind. Appl.* **45**, 772–781 (2009)
13. Jeon, S.H., Oh, K.K., Choi, J.Y.: Flux observer with online tuning of stator and rotor resistances for induction motors. *IEEE Trans. Ind. Electron.* **49**, 653–664 (2002)
14. Kenné, G., Simo, R.S., Lamnabhi-Lagarrigue, F., Arzandé, A., Vannier, J.C.: An online simplified rotor resistance estimator for induction motors. *IEEE Trans. Control Syst. Technol.* **18**, 1188–1194 (2010)
15. Marino, R., Peresada, S., Tomei, P.: On-line stator and rotor resistance estimation for induction motors. *IEEE Trans. Control Syst. Technol.* **8**, 570–579 (2000)
16. Marino, R., Tomei, P., Verrelli, C. M.: A new flux observer for induction motors with on-line identification of load torque and resistances. Paper Presented at the 18th IFAC World Congress, Milano, Italy, vol. 18, pp. 6172–6177 (2011)
17. Ticlea, A., Besançon, G.: Observer scheme for state and parameter estimation in asynchronous motors with application to speed control. *Euro. J. Control* **12**, 400–412 (2006)
18. Jadot, F., Malrait, F., Moreno-Valenzuela, J., Sepulchre, R.: Adaptive regulation of vector-controlled induction motors. *IEEE Trans. Control Syst. Technol.* **17**, 646–657 (2009)
19. Marino, R., Tomei, P., Verrelli, C. M.: Tracking control for sensorless induction motors with uncertain load torque and resistances. Paper Presented at the 8th IFAC Symposium on Nonlinear Control Systems, University of Bologna, Italy, pp. 771–776 (2010)
20. Montanari, M., Peresada, S., Tilli, A., Tonielli, A.: Speed sensorless control of induction motor based on indirect field-orientation. Paper presented at the IEEE Conference on Industrial Applications, vol. 3, pp. 1858–1865 (2000)

Chapter 2

On Visual Analytics in Plant Monitoring

Tim Tack, Alexander Maier and Oliver Niggemann

Abstract This chapter introduces methods from the field of visual analytics and machine learning which are able to handle high feature dimensions, timed systems and hybrid systems, i.e. systems comprising both discrete and continuous signals. Further, a three steps tool chain is introduced which guides the operator from the visualization of the normal behavior to the anomaly detection and also to the localization of faulty modules in production plants.

Keywords Anomaly detection · Production plant · Automation system · Visualization technique · Visual analytics

2.1 Introduction

Modern production plants grow more and more complex. A reason for this is the growing number of sensors and actuators used. Programmable Logic Controllers (PLCs) process the signals and operate the plant. Supervisory Control and Data Acquisition (SCADA) systems analyze the data provided by the PLC, to manage the process automatically. The operators activities on this level change to a passive role; from actual plant operation to plant monitoring and analysis. Due to the increasing number of signals, analyzing modern processes and detecting anomalies becomes a difficult task.

T. Tack (✉) · A. Maier · O. Niggemann
Institute Industrial IT, OWL University of Applied Sciences, Lemgo, Germany
e-mail: tim.tack@stud.hs-owl.de

A. Maier
e-mail: alexander.maier@hs-owl.de

O. Niggemann
Fraunhofer Application Center for Industrial Automation (IOSB-INA), Lemgo, Germany
e-mail: oliver.niggemann@iosb-ina.fraunhofer.de

To tackle this problem, visual analytic approaches from different scientific areas are adapted to the field of automation. As result, a novel visual anomaly detection approach is presented. It guides the operator in a top-down manner, starting from a general overview to a detailed description of identified anomalies. In this approach, visualizations of a learned reference process behavior and the currently observed one are placed side-by-side. This side-by-side visualization starts with an abstract graph computed by means of data dimensionality reduction techniques which give a coarse, time-independent system overview. The user is then guided to a more detailed visualization of the system's timing behavior. In the approach, three main ideas are combined: (i) the usage of machine learning techniques to give the operator initially an abstract view onto these complex data, (ii) the usage of machine learning techniques to visualize the normal behavior (in comparison to the current behavior) and (iii) a guided interface which leads the user step-by-step to more detailed views onto anomalous data items.

The chapter is organized as follows: In Sect. 2.2 an overview of the state of the art is given and the research gap is pointed out. Section 2.3 defines some requirements for the visualization of technical processes and introduces a new method for the visualization of high-dimensional discrete data. Based on the defined requirements, in Sect. 2.4 the visualization techniques are evaluated. For this, real data from the Lemgo Smart Factory [1] is used. To exploit the advantages found, Sect. 2.5 introduces a new plant visualization. It combines different techniques in one new approach. With it's help, a neat and informative view on the process is provided. Thus, it supports the anomaly detection performance of the operator. The results are discussed in the conclusion.

2.2 State of the Art

This section gives an overview about the state of the art and related work. In Sect. 2.2.1 some basics ideas of the visual analytics process are presented.

In Sects. 2.2.2 and 2.2.3 techniques, that can be used to visualize discrete, continuous or hybrid data (a combination of both) are described. These should support an operator with two features. At first, the high-dimensional data should be visualized in a neat way, that allows humans to deal with the overwhelming information input provided through the SCADA system. The second feature is, to enable an operator to perceive process anomalies visually.

2.2.1 Visual Analytics

In short term, visual analytics can be described as *the science of analytical reasoning facilitated by interactive visual interfaces* [2]. According to Keim [3], the visual data exploration process is organized as follows (see also Fig. 2.1):

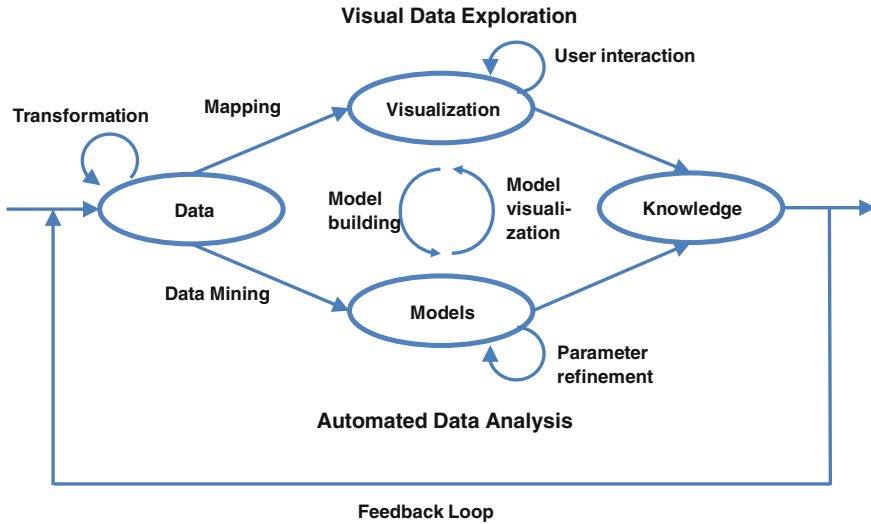


Fig. 2.1 Principle of visual analytics according to [3]

First, the data have to be acquired from the observed system. In many cases the data needs to be preprocessed (e.g. normalization or feature generation). From this, a (mathematical) model is created using data mining approaches. The model can be extended by parameter refinement. In parallel, the data are visualized for further usage. This visualization is enhanced by user interactions. Very important in this context is the tight coupling of automated and visual analysis through interaction. Both steps lead to the requested knowledge, i.e. the needed information about the systems behavior. Based on this knowledge, the operator is able to detect anomalies.

Visual analytic approaches have been applied for many years. One early example is the Londoner physician Dr. John Snow in the year 1854. To find the reason for a cholera pandemic he used a visualization method. He marked each place of occurrence in a map and was therefore able to find the reason, which was a contaminated water fountain [4].

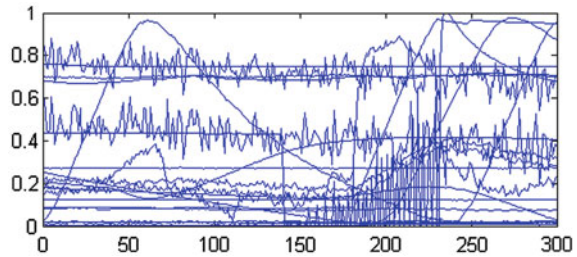
Approaches in visual analytics are considered in different research areas. For example it is used in the financial sector to visualize and analyze the fall and rise of stocks and to detect frauds, e.g. in [5]. The study of environment and climate change also often uses visualization approaches. The temperature and other relevant parameters are recorded over a long period of time. These data are visualized to recognize dependencies and to show up the changes over time. Another area of application would be the prevention of terrorist attacks [6].

There are only few examples where visual analytics has been applied to the manufacturing industry. Example Frey uses self-organizing maps to generate a two dimensional map to visualize the observed process [7]. However, there exist many approaches to create a system’s model using observations. Example in [8] a method to learn a behavior model by means of hybrid timed automata is presented. In many

Fig. 2.2 An example dataset (Figure published in [9])

time	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}
25	0	0	1	1	1	0	1	0,60	993,3	235	1,5
60	0	0	1	0	0	0	1	0,50	983,4	235	1,8
124	0	1	1	1	1	0	0	0,38	983,7	236	2,2
149	0	1	1	0	1	1	0	0,44	982,4	233	2,5
248	0	0	1	1	0	1	1	0,46	980,1	234	2,9
324	1	0	1	1	0	1	1	0,52	978,5	231	3,2
419	1	1	1	1	0	0	0	0,48	980,5	231	3,6
455	1	1	1	0	1	0	0	0,44	990,2	232	3,9
513	1	0	1	1	1	0	0	0,42	993,4	232	4,3

Fig. 2.3 Visualization with data curves (Figure published in [9])



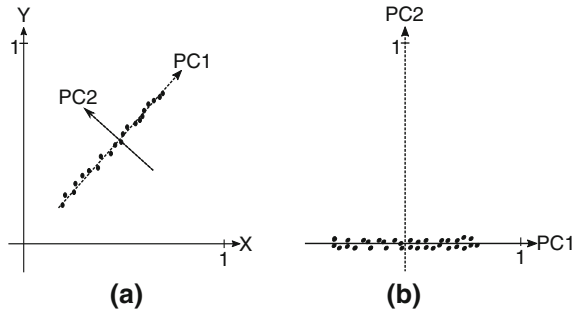
cases, especially with high amounts of data, these models are created to be used by computers and are therefore not easily accessible for humans. In this context, no appropriate method exists to visualize the plant's state to the operator. For this, special visualization methods have to be developed.

2.2.2 Visualization of Multidimensional Data

Figure 2.2 shows an excerpt taken from a process dataset. The dataset comprises a timestamp and the corresponding process variables $f_1 \dots f_{11}$. The example is rather small. Yet following the process or detecting an anomaly by viewing this figure is not easy. It can be seen that monitoring and anomaly detection in high dimensional process datasets is a tough task for computers and humans. Operators need to react on changes of large amount of different variables in different value ranges quite fast.

A trivial method to visualize data is to use signal curves in dependency of time. This simple method helps to get an overview of continuous signal trends e.g. temperature over time. Further, crossing thresholds can be seen very well. However, this method is only usable for a small number of signals using the same scaling. The visualization of many signals in one diagram leads to an information overflow, such that the single curves cannot be detected separately. Figure 2.3 shows the visualization of 30 signals with 300 data points each. Even for this small dataset the single curves cannot be separated well and it is very difficult to find an anomaly in this figure. This

Fig. 2.4 Example dataset with (a) original features and (b) principal components (Figure published in [9])



disadvantage is even worse for binary data, because the constant parts of the signals overlap and only the signal changes can be seen.

In [10] the method of parallel coordinates is introduced. This technique overcomes the problem of overlapping signal curves and allows the visualization of multiple dimensions (as coordinates) in parallel. With this, the dependencies between signals become visible. As disadvantage, it has to be mentioned, that the quality of the visualization is highly dependent on the arrangement of the features. Another method which overcomes overlapping curves in high dimensions, is the plot matrix [11]. In this, several scatter plots are depicted in a matrix such that the dependency of each pair of signals is displayed in one figure. However, very high dimensions cannot be displayed clearly as well. Example an input dimension of 20 leads to a matrix with 400 plots. An overview and comparison of the described visualization methods can also be found in [12].

2.2.3 Principal Component Analysis

As outlined in Sect. 2.2.2, it is difficult to visualize high-dimensional data. Therefore, dimensionality reduction methods have to be applied. The Principal Component Analysis (PCA) was introduced by Pearson and Hotelling and is described in the following based on [13].

The PCA finds new uncorrelated features, the principal components. The dimensionality of the dataset is then reduced by using just two principal components to describe the dataset. This is possible, because most of the variance of the original dataset, i.e. the information, is represented by the first few principal components [13]. In this contribution, a two dimensional approach is used for visualizing (choosing two principal components), because it is more difficult to extract information from a figure with three dimensions. It is impossible to create a visualization for more than three dimensions.

Figure 2.4a shows an example dataset visualized based on its two features X and Y. In Fig. 2.4b the same dataset is depicted based on its first two principal components.

We can see, that the most variance is represented by the first principal component (PC1). The variance represented by the second principal component (PC2) is rather small. In the notion of feature reduction, only PC1 would be used for data representation of the example dataset. The most information of the original dataset is preserved.

Although the most variance is kept, it must be taken into account how many information is lost due to the reduction. For example, reducing a dataset from 20 features to two principal components (reduction of 90%) while keeping 80% of the information (loss of 20%) is a quite effective way of dimensionality reduction. Nevertheless the informational loss is highly dependent on the dataset and maybe worse than in the given example. Besides the potential of dimensionality reduction it has to be considered, that the process is not visualized explicitly with respect to its time line. Further, the principal component analysis of high dimensional datasets can lead to interferences. Data and even anomalies, that can be distinguished in the original feature space, may be not distinguishable in the principal component space.

2.3 Visual Data Exploration

In this section some major requirements for the visualization of technical processes are given. In Sect. 2.3.2 a new visualization approach, the Discrete State Encoding (DSE), is introduced. It is especially developed for the visualization of high-dimensional discrete data.

2.3.1 Requirements for the Automation Domain

Every domain uses different methods to visualize the data. While climate studies use heat maps indicating the temperature, the financial industry uses curves to show trends of stocks. Visualizing automation related data, certain requirements have to be considered:

High Dimensionality. Data of production plants are typically high-dimensional. This is caused by a large amount of sensors and actuators which are used to realize production processes. Most of them are controlled by PLCs and need to be monitored by operation personnel in SCADA systems.

Different Data Types. The variety of sensors and actuators may result in different types of data. For instance a temperature sensor provides a continuous value, the temperature. Whereas a switch that activates a conveyor belt provides a discrete value, the state of the conveyor belt. Each data type sets different requirements on the visualization.

Importance of Data. Due to the high amount of data, visualizing all values would lead to an information overflow. Occurring anomalies may remain undetected. Therefore, only the most important data have to be visualized. This results in the need of methods, that distinguish between important and less important data.

Time Dependency. Processes in the automation domain are dependent on the factor time. The system's states are usually observed in relation to the process time. Therefore, the visualization approach should consider and preserve these time information. The operator should be able to assess the plant state wrt. a certain point in time.

Cyclic Processes. In typical mass production plants, process phases reoccur during the production of the same product. Therefore, the operator should be able to recognize recurring process phases as such, by examining the process visualization. As a consequence plant states, that are unusual (maybe anomalies) should be visualized in a more exposed way, to support the operator's analysis.

2.3.2 Discrete State Encoding

Since no appropriate method for the visualization of discrete data exists, this section introduces the Discrete State Encoding (DSE). It can be utilized for the visualization of datasets which consist of discrete features only. Like the PCA this technique also compresses high dimensional information. The DSE represents the plant behavior by one feature only. This new feature is then visualized over time, to provide the operator with a neat view on the plant state.

A Datasets is represented as a table with N features f_i in the columns and the measured process data, the observations, per row (see also Fig. 2.2). The DSE encodes each row of the dataset and creates a representative number, the *stateID*.

The DSE considers only discrete features, continuous features are ignored. Slightly changing continuous values would result in a new state for each observation even though the actual information has not changed significantly. The *stateID* computation is based on the following equation.

$$stateID = \sum_{i=0}^{N-1} f_{N-1-i} \cdot 2^i \quad (2.1)$$

In the next step the *stateID* values are renumbered. A serial number is assigned to each unique plant state. The *stateID* 1 is assigned to the first occurring plant state. To each newly occurring state, the next unused number is assigned, e.g. 2. To recurring states, always the same number is assigned.

Renumbering *stateIDs* is necessary to avoid bias in the visualization. Example a bit change in a highly weighted feature would affect the visualization with a higher impact than a bit change in a rather low weighted feature. This misleading perception should be avoided, following the notion of the *Lie Factor* in figures, introduced

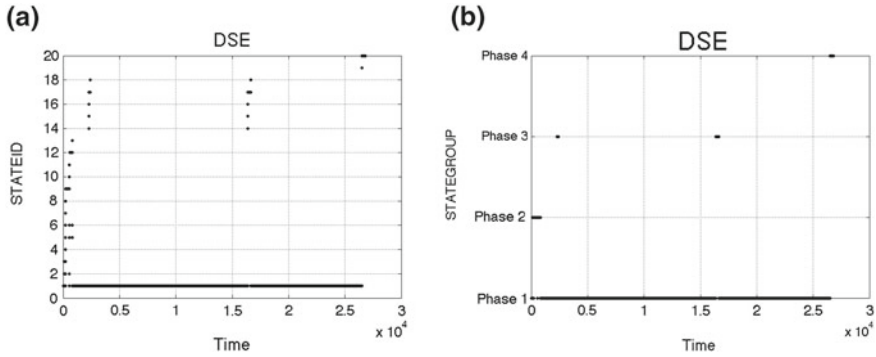


Fig. 2.5 **a** Discrete state encoding of an example dataset (Figure published in [9]) and **b** enhanced discrete state encoding of the same dataset

in [14]. In the dataset, the state change itself is the important information, not the artificial weight that is introduced for computation purpose. The renumbering preserves the state change information, but removes the bias resulting from the weights. Figure 2.5a shows a visualized discrete state encoding. The process contained more than 30 binary features and has a length of about 27,000 time units, i.e. observations.

2.4 Evaluation of Visualization Methods

In this section the visualization techniques described in Sects. 2.2 and 2.3 are evaluated. As basis, the requirements from Sect. 2.3.1 are used. While visualizing technical processes, the most important requirement is the proper visualization of the high dimensions. Since most visualization techniques (mentioned in Sect. 2.2.2) are not able to handle high dimensions properly or to reduce to the main information, only two methods are considered for detailed evaluation: The discrete state encoding (Sect. 2.4.1) and the principal component analysis (Sect. 2.4.2).

For the evaluation, a dataset from the Lemgo Smart Factory is used. The first objective is to provide an abstract process overview. The second is to detect anomalies. This is done by comparing the visualization of a reference process with the observed process, which may contain anomalies.

The observed process produces popcorn out of the resource corn. In total 19 continuous and discrete features need to be analyzed online. The production process is separated into two modules. Module one creates the product. The corn is heated until it pops. Via exhaustion the popcorn is transferred to a weight cell. In module two the popcorn is filled into cups or a larger pot, depending on what is available at time. The whole process works sequentially. First the popcorn is produced, next it is filled into the cups.

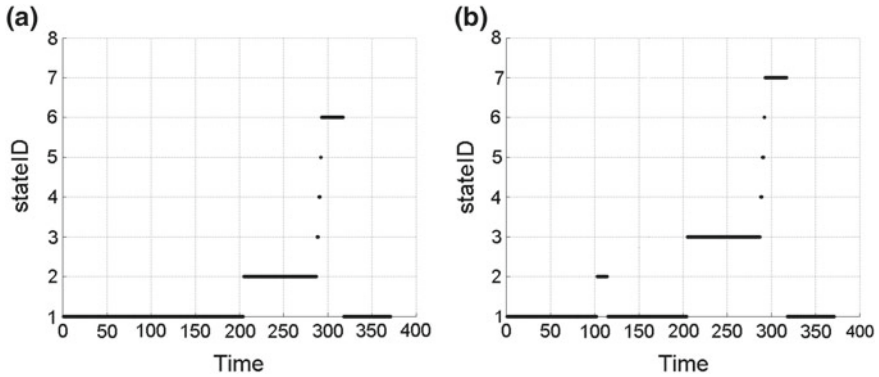


Fig. 2.6 Discrete state encoded process: **a** reference and **b** observed with failure (Figure published in [9])

2.4.1 Discrete State Encoding of a Production Process

Figure 2.6a shows the discrete encoded *stateIDs* for one process, visualized over its time line. Out of the former 19 features, one new feature, the *stateID*, is created. As mentioned before, continuous values are not taken into account to compute the *stateIDs*.

As depicted in Fig. 2.6a, the visualization provides an abstract process overview. The operator is able to analyze the process wrt. its time line. Without any expert knowledge, it can be seen that the process has three main operation phases, and some short transfer phases between time units 285–295. Plant experts confirm, that the process phases have been identified correctly. The *stateID* 1 represents the standby state of the process. In *stateID* 2 the production phase is displayed. Once enough popcorn is produced, it is filled into a cup. *stateIDs* 3–5 represent the cup filling. In *stateID* 6, the heating is turned off while the ventilation is still active to cool down the production module. Afterwards the process returns to standby (*stateID* 1).

Utilizing the visualization from Fig. 2.6a the operator is able to keep track of the process in a very convenient way. The operator is able to see the process wrt. its actual time line. Furthermore, repeating process phases are represented correctly.

In the next step, the anomaly detection performance of the DSE is tested. For that purpose, an anomaly is induced into the same dataset that has been used before. A discrete sensor (e.g. a cup filling level sensor) changes its value in an unusual moment. Figure 2.6b shows the *stateID* representation of that dataset. As depicted, the anomaly can be recognized by comparing Fig. 2.6a and b. The operator is also able to determine the point in time where the anomaly occurred. However, the operator is not able to interpret the shown anomaly in a semantic way.

Concluding, the discrete state encoding provides a neat view on the process. Further, discrete anomalies can be detected by comparing the visualizations. Even repeating process phases can be perceived easily while monitoring the *stateIDs*.

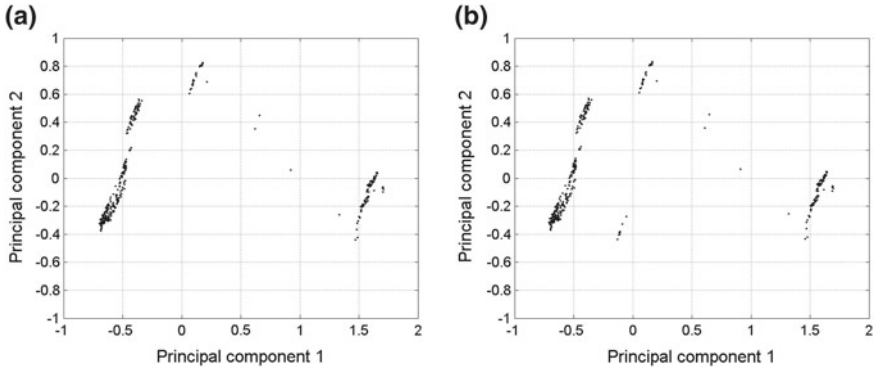


Fig. 2.7 PCA based visualization of a process: **a** reference and **b** observed with failure (Figure published in [9])

Nevertheless the operator needs some expert knowledge about the process to benefit of all information provided. A disadvantage of this visualization technique is the missing ability of visualizing continuous data.

2.4.2 Visualization of the Principal Components

In contrast to the discrete state encoding, the principal component analysis considers both continuous and discrete features for computation. In this subsection the principal component analysis is utilized to reduce the 19 features of the dataset to two representative features which are used in the visualization. The timestamp is used as an additional feature for the principal component computation. In Fig. 2.7a the process is visualized with the help of two new features, the first and second principal components. The reduction to two new features, in the given case, preserves about 80% of the variance former represented by 20 features; the informational loss is about 20%.

At first, the operator is able to see a neat process visualization. The process is grouped into three clusters. Considering the knowledge gained in Sect. 2.4.1, it can be said that this is the number of the main process phases. However, the operator is not able to semantically interpret the three clusters. It is not possible to determine whether the process phases are clustered correctly, nor to see the process phases with respect to the process time line.

To evaluate the performance in anomaly detection, an anomaly has been induced into a continuous signal. The power consumption rises without any bit change, i.e. without actively switching on a consumer. Figure 2.7b shows the visualization of the anomaly-induced process. Comparing Fig. 2.7a, b, an anomaly is perceptible. The operator is able to recognize a fourth cluster in the visualization. In addition, anomalies in discrete and hybrid features were tested. Both were visualized by this technique.

Table 2.1 Comparison of DSE, PCA and the new hybrid approach

	DSE	PCA	Hybrid approach
High dimensionality	+	o	+
Time	+	-	+
Continuous data	-	+	+
Discrete data	+	-	+
Hybrid data	-	+	+
Loss of information	+	-	+
Cyclic processes	+	+	+

In summary, the visualization based on the principal components is able to show anomalies in continuous, discrete or hybrid datasets. However, in the worst case an anomaly is not depicted by this visualization method. The reason for this can either be the lack of influence the original feature had on the principal component, the loss of information during feature reduction, or due to the interferences that are mentioned in Sect. 2.2.3. To maintain the anomaly detection performance of this visualization method in large scale datasets, expert knowledge is used to preselect significant process parts, which are used as input for the principal component analysis.

2.5 Anomaly Detection in Production Plants

The main goal of the proposed visualization approaches is to detect anomalies in the production process. In Sect. 2.5.1 a new hybrid anomaly detection approach based on visual analytics is presented. In Sect. 2.5.2 it is evaluated and some experimental results are given.

2.5.1 Hybrid Visualization and Anomaly Detection Approach

As mentioned in Sect. 2.4, both visualization techniques are able to provide a neat process overview, but still have issues in visualizing different types or special anomalies. The discrete state encoding focuses on anomalies in discrete signals and gives a process overview with respect to the time line. The principal component analysis based visualization provides a more abstract process overview and allows the viewer to detect anomalies in continuous and hybrid data. Yet, the process time line is not visualized.

Table 2.1 shows the advantages and disadvantages for both methods. It can be seen that a combination of both methods, the hybrid approach, improves the visual anomaly detection performance.

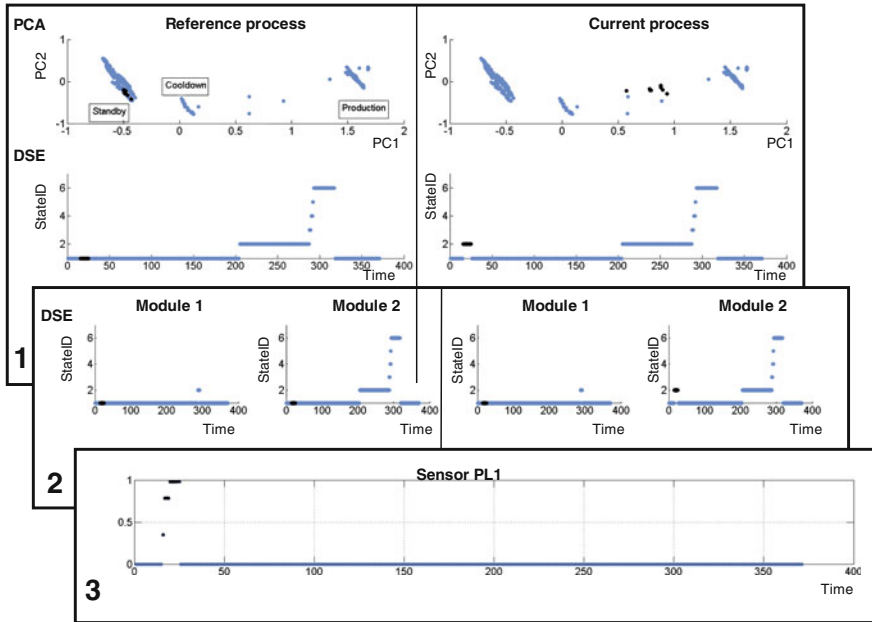


Fig. 2.8 Hybrid visualization and anomaly detection approach (Figure published in [9])

To combine the advantages of both methods, the hybrid visualization and anomaly detection approach is introduced. The method is organized in three steps. These three steps are illustrated in Fig. 2.8:

Step (1) *Observation of the process and detection of anomalies by comparing the reference process with the currently running process:*

The visualization of the principal components is used to get an abstract view on the process based on its continuous and discrete values. The discrete state encoding is used to extend the process visualization with a reference to the point in process time. Now the operator is able to compare the reference with the observed behavior in a convenient way.

Anomalies in discrete signals can be seen in the discrete state encoding, anomalies in continuous signals are displayed in the principal component visualization. To demonstrate the anomaly detection, in Fig. 2.8, an anomalous process is observed. The anomaly can be seen in both representations. In the PCA based visualization the anomalous data items form a new cluster. The discrete state encoding additionally gives the timing information: the anomaly occurred around the time stamp 20 s.

Step (2) *Determination of anomalous module:*

In this step, the process is separated based on its modules, to gain a more detailed insight. Discrete state encoding is utilized again to visualize each module separately. In this example, it can be seen that the anomaly occurred in the second module.

Step (3) Determination of anomalous signal(s):

The last element of the hybrid visualization approach refers to the continuous process values. The difference between values in a reference process and such in an anomaly induced process is calculated and shown. This allows the operator to determine which continuous sensor value differs to the process time line. Following the example in the top down manner, it can be said that the anomaly is based on an unusual energy consumption; in the signal P_{L1} .

All mentioned visualization methods are internally linked with the help of the timestamp. Based on Shneidermann's information seeking mantra *Overview first, zoom and filter, then details-on-demand* [15], the operator gets a process overview and is able to interactively explore the process. Interesting data points can be marked in one figure and corresponding data points will be highlighted in each other figures. The operator is able to see the process behavior in different levels of abstraction with respect to the time line.

In the given case, the visualization enables the operator to determine the point in time the anomaly occurred precisely. Additionally, the user is able to localize the module in which the anomaly occurred.

The combination of linkage and different visualization techniques allows the operator to find anomalies and learn about the dataset. Because of this, the PCA based visualization could be enriched with labels to provide semantic information.

2.5.2 Discussion

The hybrid visualization approach allows an enhanced visualization, since the abstract view of the principal components is combined with the temporal process visualization of the system's states. Additionally, the hybrid approach is able to handle all relevant data types for technical processes.

It was confirmed by experts, that the data abstraction using the PCA represents the normal behavior of the system. Despite these results, it is possible that not all anomalies will be displayed by the PCA based visualization. Especially in the case of high dimensional input data, important information may be unconsidered by using only the first two principal components or by the described interferences. In most cases tested, the anomalous behavior could be detected and the corresponding module and signal could be determined correctly.

Latest results in the analysis of more complex plants, larger than the Lemgo Smart Factory, showed, that the DSE based visualization can be improved further: The enhanced DSE visualization shown in Fig. 2.5b provides a more abstract view on the process with the help of *stateGroups*. One *stateGroup* represents all *stateIds* in a certain main process phase such as 'idle', 'production' or 'failure'. Expert knowledge about the process is used, to determine the *stateGroups* and to assign particular *stateIDs* to its group.

A minor disadvantage of the proposed approach is that expert knowledge is still needed to analyze the plant's behavior in detail. Nonetheless, it is possible to detect anomalies and the anomalous production module(s) and signal(s) without any expert knowledge, by assessing the visualization.

Another disadvantage is, that the proposed approach works well for a cyclic process, but not necessarily for extended production plants which deal with different variants of products. This will be improved in future work.

2.6 Conclusions

In this chapter a novel visual analytics approach for the visualization of technical processes is presented. The discrete state encoding gives a neat overview of the observed process and shows the main process states over the time line. The principal component analysis gives a more abstract overview of the process and additionally includes continuous data. Both methods were connected to combine their advantages.

Further, it is shown how the visualization and anomaly detection approach can be used to analyze a technical process. In three steps the operator is guided through the observation of the current behavior and the corresponding reference behavior. This side-by-side visualization allows operators to detect anomalies visually. Different abstraction levels support a detailed process analysis, by zooming into module or even signal level. Thus, the operator is guided step by step to the signal, that caused the anomaly.

In further work some other visualization approaches will be explored. These shall show the most relevant data in a more intuitive way to give the possibility to analyze the process behavior without (or at least with less) expert knowledge. To face the disadvantage of the DSE, continuous values can be discretized using an n-bit-discretization. This will also be considered in future work.

Furthermore, the reference visualization will consider more than only one process. This will provide a more generalized view on the plant's behavior.

Acknowledgments This work is an extension of the chapter titled *Visual Anomaly Detection in Production Plants* by Alexander Maier, Tim Tack and Oliver Niggemann, published in *9th International Conference on Informatics in Control, Automation and Robotics (ICINCO 2012), Rome, Italy*.

References

1. Institute Industrial IT: Lemgo Smart Factory. <http://www.smartfactory-owl.de> (2012)
2. Thomas, J.J., Cook, K.A. (eds.): *Illuminating the Path: The Research and Development Agenda for Visual Analytics*. IEEE Computer Society Press, Los Alamitos (2005)
3. Keim, D.A., Kohlhammer, J., Ellis, G., Mansmann, F. (eds.): *Mastering the Information Age—Solving Problems with Visual Analytics*. Eurographics, Leipzig (2010)

4. Tufte, E.R.: *Visual Explanations: Images and Quantities, Evidence and Narrative*. Graphics Press LLC, Cheshire (1997)
5. Huang, M.L., Liang, J., Nguyen, Q.V.: A visualization approach for frauds detection in financial market. In: *Proceedings of the: 13th International Conference Information Visualisation. IV '09*. Washington, USA, IEEE Computer Society (2009) 197–202
6. Thomas, J.J., Cook, K.A.: A visual analytics agenda. *IEEE Comput. Graphics Appl.* **26**, 10–13 (2006)
7. Frey, C.W.: Diagnosis and monitoring of complex industrial processes based on self-organizing maps and watershed transformations. In: *IEEE International Conference on Computational Intelligence for Measurement Systems and Applications* (2008)
8. Niggemann, O., Stein, B., Vodenčarević, A., Maier, A., Kleine Büning, H.: Learning behavior models for hybrid timed systems. In: *Twenty-Sixth Conference on Artificial Intelligence (AAAI-12)*, Toronto, Ontario, Canada (2012) 1083–1090
9. Maier, A., Tack, T., Niggemann, O.: Visual anomaly detection in production plants. In: *International Conference on Informatics in Control, Automation and Robotics, Rome, Italy* (2012)
10. Inselberg, A.: The plane with parallel coordinates. *Visual Comput.* **1**, 69–91 (1985)
11. Cleveland, W.: *Visualizing Data*. AT&T Bell Laboratories, Murray Hill (1993)
12. Ward, M., Grinstein, G., Keim, D.A.: *Interactive data visualization: foundations, techniques, and application*. A. K. Peters Ltd., ISBN: 978-1-56881-473-5. <http://www.idvbook.com> (2000). Accessed May 2010
13. Jolliffe, I.T.: *Principal Component Analysis*. Springer, New York (2002)
14. Tufte, E.: *The Visual Display of Quantitative Information*. Graphics Press, Cheshire (2001)
15. Shneiderman, B.: The eyes have it: a task by data type taxonomy for information visualizations. In: *IEEE Symposium on Visual Languages*, p. 336 (1996)

Chapter 3

Global Optimization for 2D SLAM Problem

Usman Qayyum and Jonghyuk Kim

Abstract A globally optimal approach is proposed in this work for map-joining SLAM problem. Traditionally local optimization based approaches are adapted for SLAM problem but due to highly non-convex nature of the SLAM problem, they are susceptible to local minima. In this work, we have exploited the theoretical limit on the number of local minima. The proposed approach is not dependent upon the good initial guess whereas existing approaches in SLAM literature requires a good starting point for convergence to the basin of global minima. Simulation and real dataset results are provided to validate the robustness of the approach to converge to global minima. This chapter provides the robotics community to look into the SLAM problem with global optimization approach by guarantying the global optimal solution in a least square cost function particularly when covariance matrices are defined as spherical.

Keywords Local minima · Map-joining · Gauss-Newton optimization · Greedy random adaptive search procedure · Optimal solution

3.1 Introduction

Simultaneous Localization and Mapping (SLAM) has drawn significant interests in robotics communities, as it enables the robotic vehicles to be deployed in a fully autonomous way for various applications. SLAM literature is mostly categorized into

U. Qayyum (✉) · J. Kim
School of Engineering, Australian National University, Canberra, ACT 0200, Australia
e-mail: usman.qayyum@anu.edu.au

J. Kim
e-mail: jonghyuk.kim@anu.edu.au

two main streams: Filtering based [1] and Maximum likelihood based [2] approaches. The first stream consists of Extended Kalman filter and Information filter [3] which requires linearization of process and measurement models with a cost of potential divergence and inconsistency. FastSLAM [4] is based upon factorization of posterior but, due to limited numbers of particles, it is unable to represent the trajectory posterior in the long run.

In graph-based SLAM approaches measurements acquired during robot motions are modeled as constraints. The goal of these approaches is to estimate the configuration of parameters that maximally explain a set of measurements affected by Gaussian noise (minimizes the nonlinear least square error). The pioneering work in graph-based SLAM is by [2] in which brute force technique for range scan alignment was proposed. With the assumption of known rotation [5] introduced a Gauss-Seidel relaxation. The work was improved by [6] solving a network at different level of resolution [7]. Came up with QR factorization of information matrix to solve the full SLAM problem.

Stochastic gradient descent (SGD) was used by [8] to solve the pose only SLAM problem by addressing each constraint individually and surprises many researcher as the algorithm can converge to the correct solution with poor initial values. Recent research [9] has focused on making these algorithms more efficient and robust showing that its online implementation is feasible. Although these approaches perform efficiently in practice, very little attention has been paid on the convergence condition and none of them can guarantee a global minimum over different initial guesses. Figure 3.1 shows the results of a Gauss-Newton approach on publicly available dataset [10] with different random initial guesses, resulting in different local solutions.

Global optimal solutions to highly nonlinear problems has been shown to NP-hard [11]. In structure from motion research, the guaranteed global optimal solution is investigated with known rotation framework for L_∞ [12] and branch and bound based approaches [13] whereas in our work no such assumption of known rotation is considered (typically the nonlinearity in measurements of mobile robot applications is due to robot orientation). In recent work [14] proposed a swarm optimization based approach to estimate (almost) optimal maps. Their work is based upon meta-heuristic optimization approach which is similar to our work but they present map as a tree of fragments/maps with particle filtered based sampling approach and finally conducted an ant colony search to obtain (almost) optimal solution.

In this chapter we provide an approach to get global optimal solution for SLAM problem in map-joining problem with spherical covariance, which has never been proposed before to the best of author's knowledge. Feature based SLAM problem is decomposed as a problem of joining submaps. Necessary and sufficient condition for the existence of at most two local minima [15] is exploited by a meta-heuristic approach called GRASP (greedy randomized adaptivesearch proce-

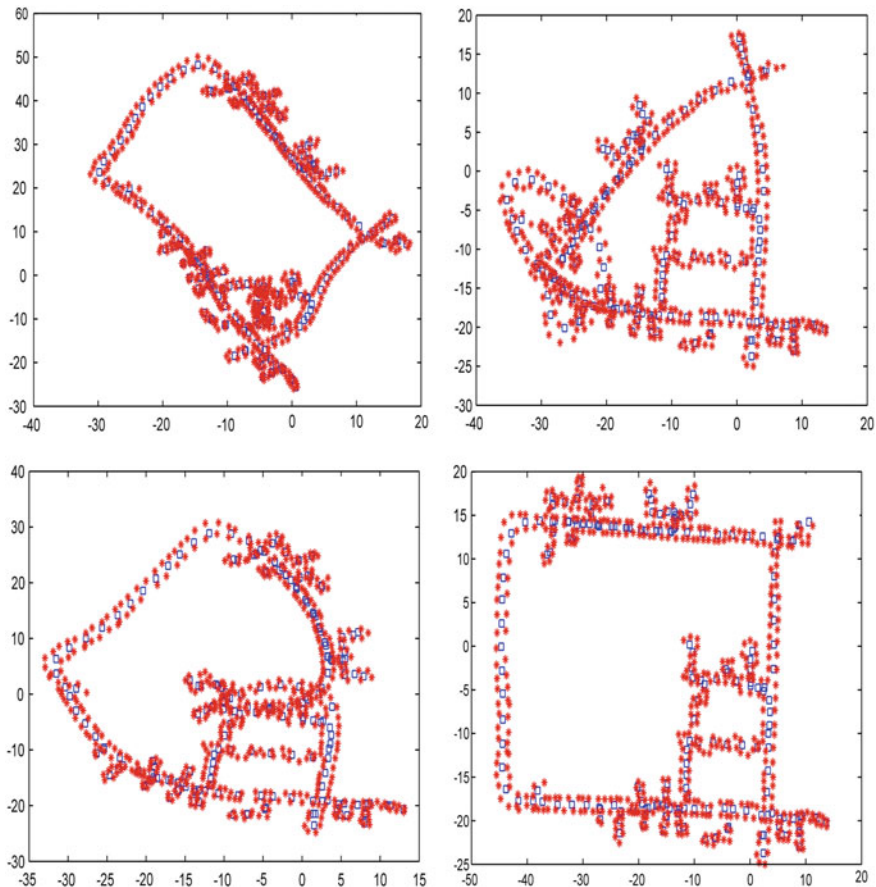


Fig. 3.1 Examples of local solution for DLR dataset [10] with different random initial guess

ture) [16] which is combinatorial optimization to obtain global optimal solution. Meta-heuristic approaches optimize by iteratively refining the candidate solution by combining randomness with local search methods [17]. Unknown landmark positions and vehicle pose are considered as initial guess in a planar environment (3DOF case).

The outline of this chapter is as follows: Sects. 3.2 and 3.3 will provide nonlinear least square formulation and number of local minima in SLAM problem. Sections 3.4 and 3.5 will provide detailed discussions on nonlinear global optimization based approach and greedy search strategy. Section 3.6 will briefly explain global optimal approach to map-joining. Results and discussions will be presented in Sect. 3.7 followed by conclusion.

3.2 Nonlinear Least Square Formulation

The dimension of the SLAM problem is very high when it is formulated as a nonlinear least square [18] because all vehicle poses and feature locations are considered as parameters to be determined. The decomposition of SLAM problem into submaps not only helps to reduce the computational complexity but also helps to improve consistency by decreasing the nonlinearity of the system [3, 19]. The assumption we undertake in this research is that, every SLAM problem can be decomposed into local maps and then solved for global optimal solution. The relative relation of local maps has fluid behavior whereas the internal structure of each local map is well known and can be optimized independently with respect to local coordinate [18]. The nonlinear least square formulation for local map joining is to minimize an objective function as follows:

$$F(x) = \arg \min_x \sum (\|E_p\|_U^2 + \|E_f\|_V^2), \quad (3.1)$$

where the state vector is $x = \{X, M\}$ with X being the poses (position and orientation) of local maps and M the features positions in absolute coordinate frame. U and V are corresponding covariance matrices of pose and feature observations respectively. The poses are composed of $\{p_1, \varphi_1, \dots, p_l, \varphi_l\}$ where the end pose of each local map is the start pose of next local map. The N map features are defined as $M = \{f_1, \dots, f_N\}$. SLAM recasted as a map-joining problem is shown in Fig. 3.2.

Let there be a local map l defined as $X^l = \{p^l, \varphi^l\}$ with n features $M^l = \{f_1^l, \dots, f_n^l\}$ present in the local coordinate frame. The local map pose X^l is the observation of relative pose of p^l, φ^l from the global state vector X pose p_{l-1}, φ_{l-1} as

$$E_p = X^l - H_{odo}(X), \quad (3.2)$$

and the observation model for odometry is defined as

$$H_{odo}(X) = \begin{bmatrix} R(\varphi_{l-1})^T(p_l - p_{l-1}) \\ \varphi_{l-1} - \varphi_l \end{bmatrix}. \quad (3.3)$$

The $SO(2)$ rotation matrix is defined as

$$R(\varphi) = \begin{bmatrix} \cos(\varphi) & -\sin(\varphi) \\ \sin(\varphi) & \cos(\varphi) \end{bmatrix}. \quad (3.4)$$

The feature positions $M^l = \{f_1^l, \dots, f_n^l\}$ in local map l are observation of relative position of features and are related (assumed data association) to global state vector. The relative feature position error E_f is defined as

$$E_f = M^l - H_{feat}(X, M), \quad (3.5)$$

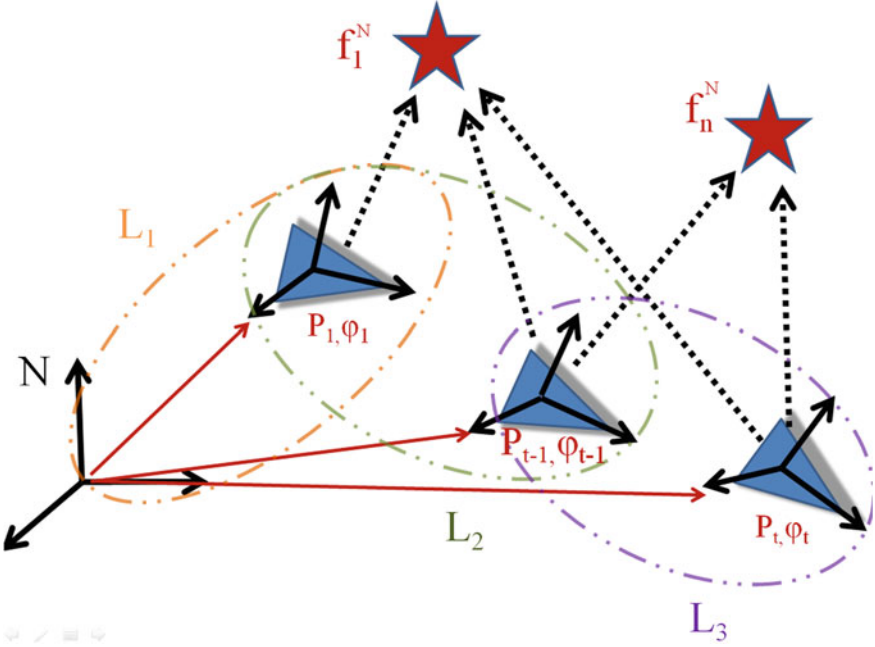


Fig. 3.2 SLAM is recasted as a map-joining problem: One is a growing global map (navigation coordinate with *solid lines*) and the other is a new local map L (represented with *ellipses*). The *dashed lines* indicate the odometry and feature observations

whereas the observation model for relative position of features is a function of M and X

$$H_{feat}(X, M) = \begin{bmatrix} R(\varphi_{l-1})^T (f_{l1} - p_{l-1}) \\ \vdots \\ R(\varphi_{l-1})^T (f_{ln} - p_{l-1}) \end{bmatrix}. \quad (3.6)$$

The Mahalanobis distance for both relative errors in the cost function with zero-mean Gaussian noise with covariance U, V can be written as

$$F(x) = \sum (X^l - H_{odo}(X))^T U^{-1} (X^l - H_{odo}(X)) + \sum (M^l - H_{feat}(X, M))^T V^{-1} (M^l - H_{feat}(X, M)). \quad (3.7)$$

The measurement and process models are both nonlinear functions, and thus the nonlinear objective function is linearized multiple times to reach local minima. Generally local approaches solve the objective function $F(x)$ as

$$F(x + \delta x) = F(x) + J\delta x, \quad (3.8)$$

where $J = \partial F(x)/\partial(x)$. Let $\varpi = (F(x + \delta x) - F(x))$, then it becomes

$$J\delta x = \varpi, \quad (3.9)$$

The solution can be found using pseudo-inverse of J

$$\begin{aligned} J^T J \delta x &= J^T \varpi \\ \delta x &= (J^T J)^{-1} J^T \varpi \end{aligned} \quad (3.10)$$

By including the covariance estimates (U, V) as ξ , Eq. 3.10 becomes

$$\delta x = (J^T \xi^{-1} J)^{-1} (J^T \xi^{-1}) \varpi. \quad (3.11)$$

When the Eq. 3.11 is solved only ones by an optimization based approach i.e. [7], it yields a similar result like the EKF or Extended information filter [1]. The advantage in optimization based approach comes with repeatability of solution for Eq. 3.11 with different linearization points.

3.3 Number of Local Minima in MAP Joining SLAM

The SLAM formulation as linearized version of nonlinear objective function assumes local convexity, that is with a reasonable initial estimate (near the global basin of attraction), the algorithm will converge to global minima. SLAM is an incremental process and at each step, small number of new parameters need to be estimated. However when the odometry and feature observation are not consistent with each other, then the local optimizer can struck in local minima.

A close lookup at the observation model in Eqs. 3.3 and 3.6 reveals that nonlinearity is only due to orientation. Recently [15] provides a theoretical bound on local minima in map-joining SLAM problem, by deriving a nonlinear equation depending only on orientation error under the assumption that covariance matrices are spherical matrices. The research findings in their work suggest at most two and at least one local minima can occur. The approach, proposed in this chapter, exploits the upper theoretical bound under noisy observations to obtain global minima by a meta-heuristic approach (discussed in following section) hence obtaining a global optimal solution.

3.4 Greedy Random Adaptive Search Procedure

GRASP is a multi-start meta-heuristic approach to solve combinatorial problems [16]. Previously, it has been successfully deployed in traveling sales man problem and firstly proposed here for SLAM problem. GRASP basically consist of two phases:

local search and feasible solution construction. The construction phase builds a feasible solutions (using greedy approach), whose neighborhood is searched by a local search phase to find local optima. By using different feasible solutions as starting points for local search in a multi-start procedure will usually lead to good, though, most often, suboptimal solutions. While in our problem at worst case, we encountered two local minima when joining two local maps, so the upper bound is searched by multi-start until global optimal solution is returned, which is the optimal solution. The pseudo code in Algorithm 1, details the working of GRASP approach, where local search is performed by the gauss-newton formulation of Eq. 3.11.

Algorithm 1: GRASP-Algorithm: Determination of Optimal Solution.

```

inputs : observations =  $\{X^l, M^l\}$ ,  $x = \{pose, landmark\}$ 
 $x^* = \{\}$ 
while check local minima condition or max iteration are not reached do
   $x \leftarrow RandomizedGreedyAlgo(.) \rightarrow Algo2$ 
   $x \leftarrow LocalSearch(x, observations)$ 
  if ( $f(x) < f(x^*)$ ) then
     $x^* = x$ 
  end if
end while
return  $x^*$ 

```

3.5 Randomized Greedy Algorithm

We proposed long-term memory based greedy algorithm to determine a feasible solution. Figure 3.3 (left) details a simple example on 1D in which at most two local minima are considered. The two feasible solutions (x_1, x_2) are generated by GRASP approach and among them two minima are found whereas x_1 reveals the global optimal solution x^* . The generation of feasible solution is the key to obtain the optimal solution, in timely manner from a high dimensional search space of parameters. Figure 3.3 (right) describes a search space reduction mechanism in which the initial guess x_1 is first hypothesized, which determines a local optimal x^* (the intermediate traversal solution of the local optimizer are stored in long-term memory for search space x_1 to x^*). The next hypothesis of initial guess is being made by greedy algorithm in consideration with the already traversed solution space in long-term memory (the goal is to avoid making an initial guess from already traversed space). The selection of new initial guess is based upon absolute distance criteria (greedy approach), in which the new initial guess for local optimizer is not from the already considered search space. The greedy algorithm helps to reduce the search space and improves the time efficiency as against the exhaustive bruteforce search in solution

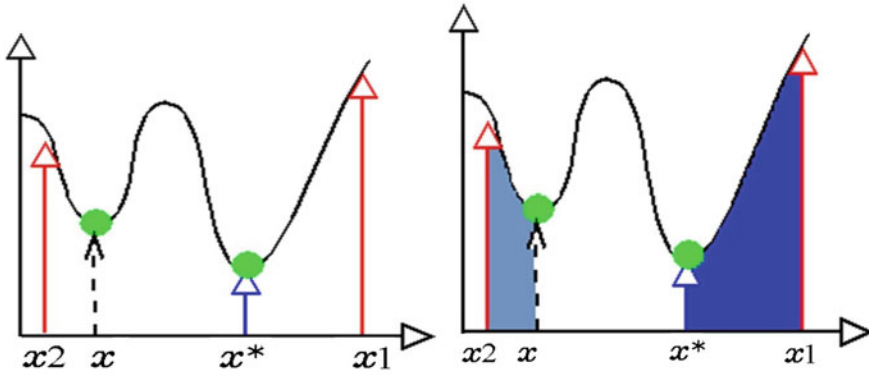


Fig. 3.3 1D problem with 2 local minima revealing possible feasible solutions from local optimization

space. The pseudo-code in Algorithm 2, describes the greedy algorithm, in which the long-term memory of all the solution space is maintained. The selection of new feasible solution is determined similar to 1D explained approach on orientation space (as vehicle and feature positions are linear with respect to orientation). Selection of ϵ in radians decides the selection of new hypothesis for initial guess to boot the local optimizer.

An appealing characteristic of GRASP approach is the ease of implementation by setting and tuning few parameters. The computation time of the approach does not vary much from iteration to iteration and increases linearly with the number of iterations whereas the time increases combinatorially with the increase of searched spaced parameters.

Algorithm 2: GRASP-Algorithm: Randomized greedy algorithm.

```

while Initial guess not found | Max iter not reached do
    x = random(.)
    found = 1
    for i=1:num of elements in LTmemoryX do
        Oldx = LTmemoryX[i]
        if |(x - Oldx)| <  $\epsilon$  then
            found = 0
            Break  $\rightarrow$  select another x
        end if
    end for
end while
return x
    
```

3.6 Global Optimal Solution to Map-Joining

Most of the map joining approaches start with linear approximation of map states and do optimization as a post processing step to estimate the local solution [3, 18, 19]. Our approach is not dependent upon the initial guess and bound to search the optimal results by modified GRASP approach. We initialized each map randomly (unknown feature and vehicle pose) and solve the map joining problem as illustrated in pseudo code Algorithm 3. The data association is assumed to be known. Two maps are considered at each time and provided to GRASP algorithm, which returns the optimal solution for those two sets of maps observation. The sequential joining is not necessary and parallel algorithm can be employed for faster computation, if the running time is bottleneck in a large scale environment. The input to Algorithm 1 will be set of observations (relative feature and odometry information of local maps l) and global map state vector X, M to obtain x^* .

3.7 Results and Discussion

The performance of the proposed algorithm is tested on simulated [19] as well on real dataset [10], which are both available publicly.

Algorithm 3: MAP Joining Optimal: GRASP variant for Map Joining.

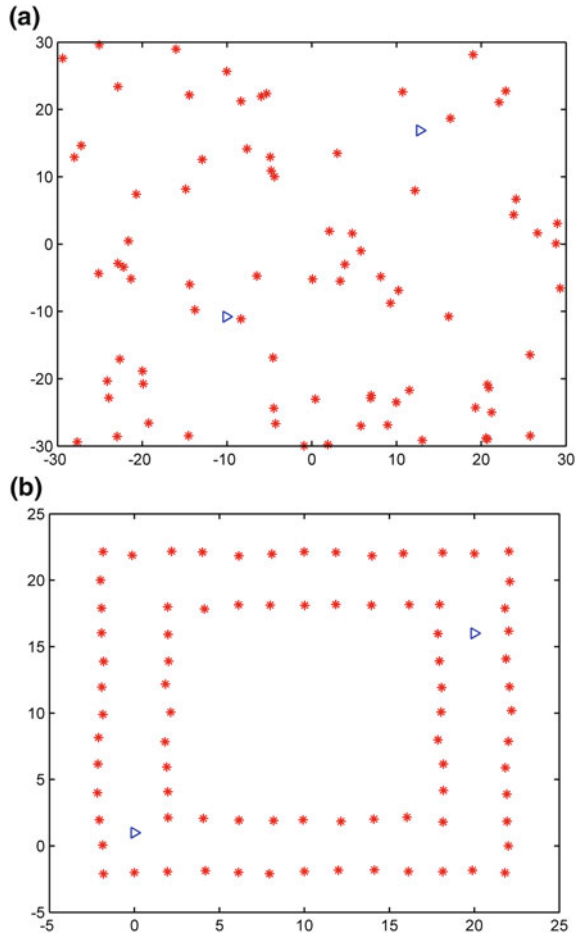
```

x = first local map
while Fuse local map k+1( $l$ ) into x do
    Data Association assumed to be known
    Initialize random pose/landmark for  $l$  into x
    x= GRASP-Algorithm (x, Local Map  $l$ )
end while
return ( $x^*=x$ )

```

The $150 \times 150 \text{ m}^2$ simulation environment [19] containing 2,500 features uniformly space in rows and columns is tested first. The robot started from the left bottom corner of the square and followed a big loop. A sensor with a field of view of 180° and a range of 6m was simulated to generate relative range and bearing measurements between the robot and the features. The dataset is divided into two local maps with unknown initial guess (covariance matrices are set to identity). Multiple trials were performed with random initial guesses to be processed with the proposed approach for map-joining, which always converge to global optimal solution without being affected by variation in initial guess, as shown in Fig. 3.4. The approach was also tested on a multiple local maps (as the upper bound is available for joining two individual local maps), so the obtained solution is an approximation of the original problem. 50 local maps in total with 612 observed features and 1,374 odometry/feature measurements were made from the robot poses. Figure 3.5a shows

Fig. 3.4 GRASP based map-joining for simulation dataset [19] with 2 local maps. **a** Random initial guess with two local maps, **b** GRASP based map-joining



the intermediate results of 25th local map where new local map with random initial guesses of landmark position and pose. The GRASP based smoothing is performed and approximated optimal results obtained is shown in Fig. 3.5b. Final results with ground truth result is shown in Fig. 3.6, showing estimated position against the interpolated ground truth positions. The GRASP based approach with unknown initial guess is tested on DLR dataset [10] and compared with the state of the art map-joining approach [19]. DLR dataset is acquired with a camera attached on a wheeled robot and odometry. The robot moved around a building detecting scattered artificial white/black landmarks, placed on the ground. Odometry measurements and relative position of the observed landmarks are being provided. This dataset is also divided into two local maps. Figure 3.7 shows the results of the global optimal solution obtained after joining local maps with unknown initial guess (multiple trials resulted in the same optimal solution hence showing the independence to good initial guess).

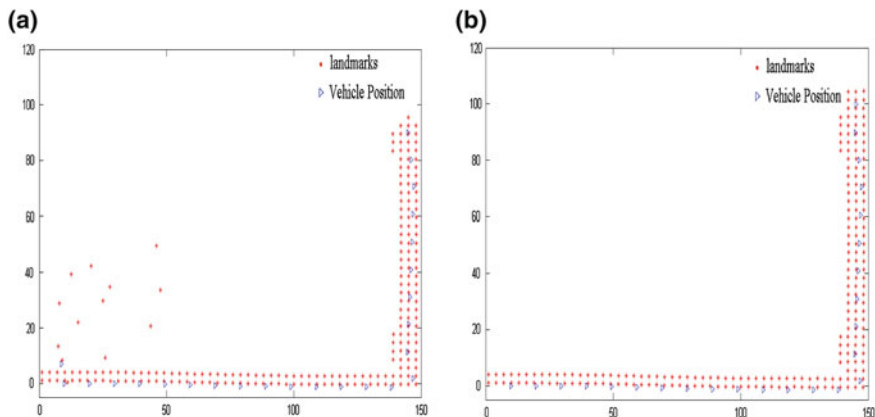


Fig. 3.5 Intermediate results of simulation dataset [19] with fifty local maps. **a** Before joining, **b** After joining

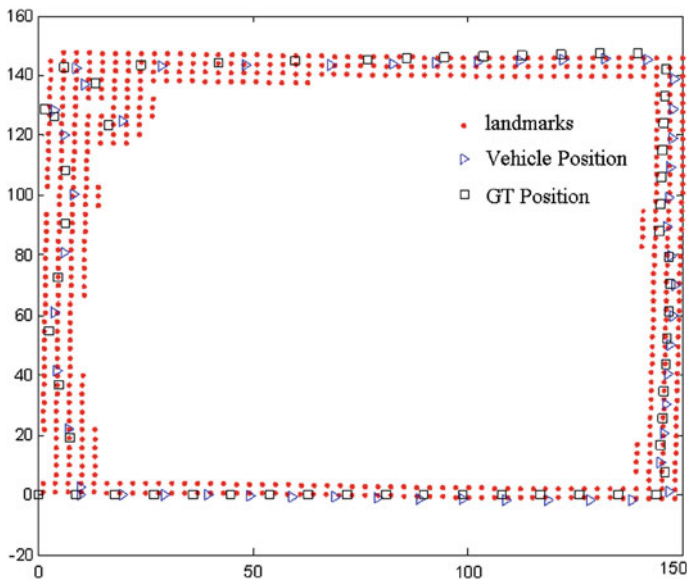


Fig. 3.6 Ground truth and estimated vehicle/landmark positions for simulation dataset [19] with 50 local maps

The proposed approach is also tested on 200 local maps with 540 observed features and 1,680 odometry/feature observations is considered with known data association. Figure 3.8a shows the random initial guess for vehicle pose and feature positions in global frame of reference, to be processed by GRASP approach. Figure 3.8b, c shows a visual comparison of proposed approach with [19] (which is booted with linear initialization whereas our proposed approach is not dependent upon known initial guess).

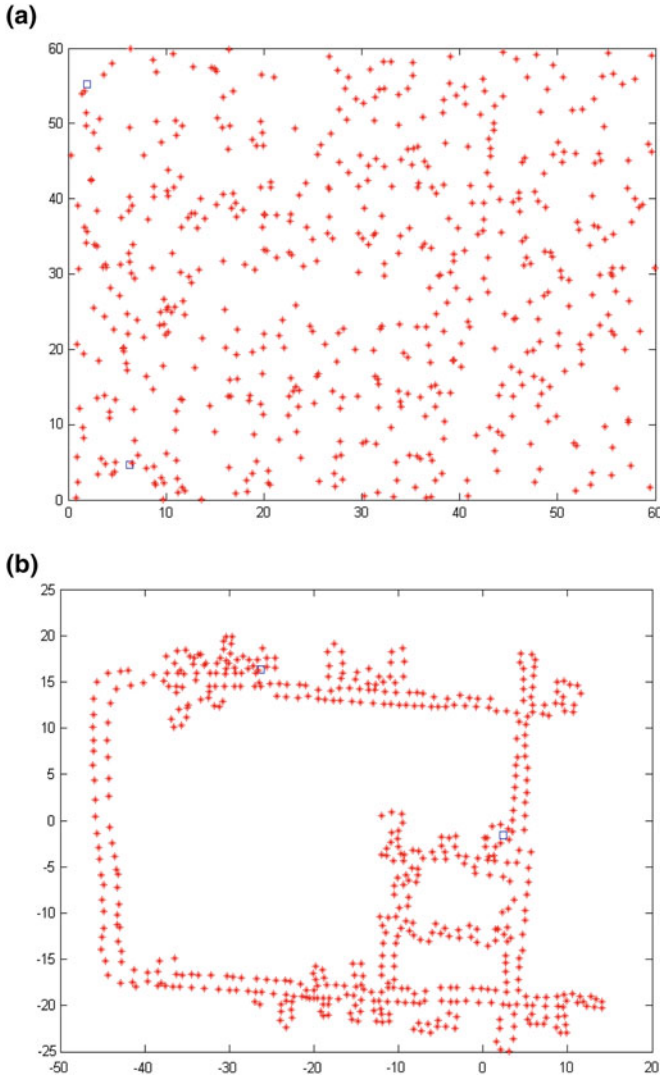
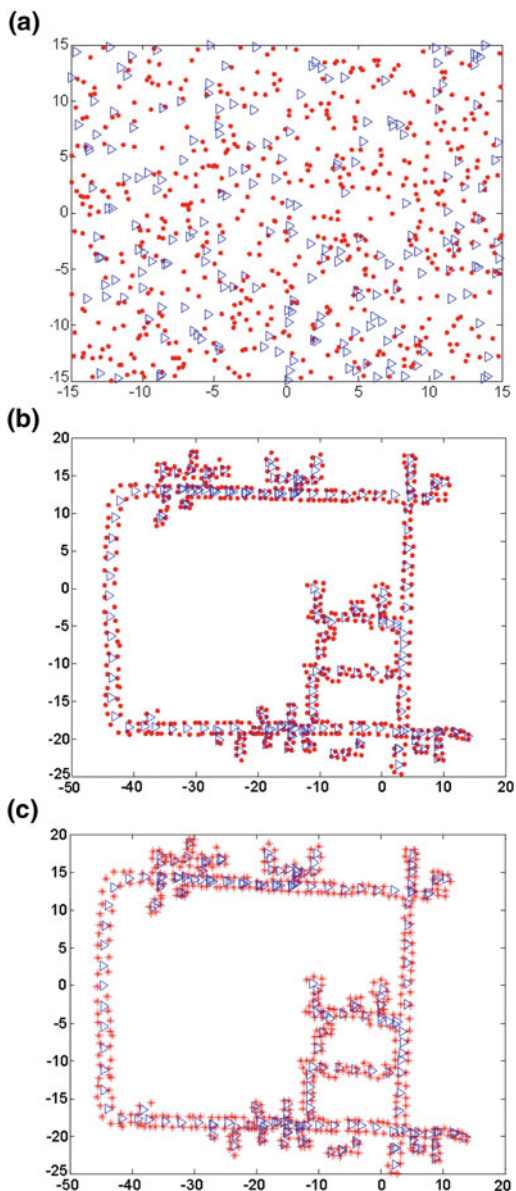


Fig. 3.7 GRASP based map-joining for simulation dataset [10] with 2 local maps. **a** Random initial guess with two local maps, **b** GRASP based map-joining

The performance of the experimental results validates the proposed idea and makes the SLAM problem solvable by global optimization based approach without initial guess.

Fig. 3.8 DLR dataset results using GRASP with 200 local maps, showing global convergence **(b)** with random initial guesses **(a)**. **a** Random initial guess for local maps. **b** GRASP based vehicle/landmark position estimates. **c** I-SLSJF with EIF based initialization [19]



3.8 Conclusions and Future Works

A practical approach for finding the globally optimal solution to SLAM is presented. Local optimization based strategies which are mostly adopted for SLAM problem are highly susceptible to local minima problem due to non-convex structure of the

problem. By exploiting the theoretical limit on the number of local minima, we proposed a framework to estimate a global optima. The proposed approach is not reliant on good initial guess which is the primary condition of local optimization based approaches for global convergence. Experimental results are provided on different datasets available online to validate the robustness of approach.

Future work will provide the time/memory comparison of the proposed approach and an extension to 3D SLAM problem.

Acknowledgments This work is supported by the ARC DP Project (DP0987829) funded by the Australian Research Council (ARC). We are also thankful to [10] and [19] for open source implementation and datasets.

References

1. Bailey, T., Durrant-Whyte, H.: Simultaneous localization and mapping (SLAM): Part II. *IEEE Robot. Autom. Mag.* **13**(3), 108–117 (2006)
2. Lu, F., Milius, E.: Globally consistent range scan alignment for environment mapping, *J. Auton. Robots* **4**, 333–349 (1997)
3. Huang, S., Wang, Z., Dissanayake, G.: Sparse local submap joining filter for building large-scale maps. *IEEE Trans. Robot.* **24**(5), 1121–1130 (2008)
4. Doucet, A., Freitas, N., Murphy, K., Russel, S.: Blackwellized particle filtering for dynamic bayesian networks. In: *International Conference on Uncertainty in Artificial Intelligence*, p. 17618 (2002)
5. Duckett, T., Marsland, S. Shapiro, J.: Fast, on-line learning of globally consistent map. *J. Global Optim.* **12**(3), 287–300 (2002)
6. Frese, U., Larsson, P., Duckett, T.: A multilevel relaxation algorithm for simultaneous localization and mapping. *IEEE Trans. Robot.* **21**(2), 196–207 (2005)
7. Dellaert, F., Kaess, M.: Square root SAM: Simultaneous localization and mapping via square root information smoothing. *Int. J. Robot. Res.* **25**(12), 1181–1203 (2006)
8. Olson, E., Leonard, J., Teller, S.: Fast iterative optimization of pose graphs with poor initial estimates. In: *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2262–2269 (2006)
9. Kummerle, R., Grisetti, G., Strasdat, H., Konolige, K., Burgard, W.: g2o: A general framework for graph optimization. In: *IEEE International Conference on Robotics and Automation (ICRA)*, (2011)
10. Kurlbaum, J., Frese, U.: A benchmark data set for data association, Technical Report, University of Bremen, Data Available on <http://cres.usc.edu/radishrepository/>, (2010)
11. Freund, R., Jarr, F.: Solving the sum-of-ratios problem by an interior-point method. *J. Global Optim.* **19**(1), 83–102 (2001)
12. Astrom, K., Enqvist, O., Olsson, C., Kahl, F., Hartley, R.: An L_∞ approach to structure and motion problems in 1D-vision. In: *International Conference on Computer Vision (ICCV)* (2007)
13. Olsson, C., Kahl, F., Oskarsson, M.: Branch and bound methods for euclidean registration problems. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2008)
14. Iser, R., Wahl, F.: AntSLAM: global map optimization using swarm intelligence. In: *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 265–272 (2010)
15. Huang, S., Wang, H., Frese, U., Dissanayake, G.: On the number of local minima to the point feature based SLAM problem. In: *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 265–272 (2012)
16. Resende, M., Ribeiro, C.: Greedy randomized adaptive search procedures. In: *Handbook of Metaheuristics*, pp. 219–249 (2003)

17. Blum, C., Roli, A.: Metaheuristics in combinatorial optimization: overview and conceptual comparison. *ACM Comput. Surv.* **35**(3), 268–308 (2003)
18. Ni, K., Steedly, D., Dellaert, F.: Tectonic sam: exact, out-of-core, submap-based slam. In: *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1678–1685 (2007)
19. Huang, S., Wang, Z., Dissanayake, G., Frese, U.: Iterated SLSJF: A sparse local submap joining algorithm with improved consistency. In: *Australasian Conference on Robotics and Automation (ACRA)* (2008)

Chapter 4

Stochastic Models and Optimization Algorithms for Decision Support in Spacecraft Control Systems Preliminary Design

Eugene Semenkin and Maria Semenkina

Abstract Technological and command-programming control contours of spacecraft are modelled with Markov chains. These models are used for the preliminary design of spacecraft control system effective structure with the use of special DSS. Corresponding optimization problems with algorithmically given functions of mixed variables are solved with a special stochastic algorithm called self-configuring genetic algorithm that requires no settings determination and parameter tuning. The high performance of the suggested algorithm is proved by the solving real problems of the control contours structure preliminary design.

Keywords Spacecraft control contours modelling · Markov chains · Effective variant choice · Complex optimization problems · Self-configuring genetic algorithm · Island model

4.1 Introduction

Current efforts by the developers of spacecraft are concentrated upon increasing the usage effectiveness of existing spacecraft systems and improving the development and design process for new ones. One of the ways to achieve these aims is a rational choice of the effective variants of the developing systems. This requires the application of adequate models, effective algorithmic tools and powerful computers. This application will allow multivariant analysis of the developing systems that is currently not so easy due to their complexity.

E. Semenkin (✉) · M. Semenkina
Institute of Computer Sciences and Telecommunication, Siberian State Aerospace University,
Krasnoyarskiy Rabochiy ave., 31, 660014 Krasnoyarsk, Russia
e-mail: eugenesemenkin@yandex.ru

M. Semenkina
e-mail: semenkina88@mail.ru

One of the most difficult and under developed problems is that of the synthesis of a spacecraft's control systems. These are currently solved with more empirical methods rather than formalized mathematical tools. Usually, the spacecraft control system design is a sophisticated process involving the cooperation of numerous experts and departments each having their own objectives and constraints. Nevertheless, it is possible to mathematically model some subproblems and to obtain some qualitative results of computations and tendencies that could provide interesting information for experts. The usual position of system analyst in such a situation is as mediator for high level decision making, dealing with informal problems for which it is impossible to develop a mathematical model, and lower level computations for which strong mathematical models exist but the results of them do not always match. If, in this intermediate position when mathematical models are strong enough but very complicated for analysis, we intend to implement a decision support system for the choice of effective variants then we have to realize that the optimization problems arise here are intractable for the majority of known algorithms.

We suggest modelling the functioning process of a spacecraft's control subsystems with Markov chains. We explain the modelling with small models and then give illustration of large models that are closer to real system. The problem of choosing an effective variant for a spacecraft's control system is formulated as a multi-scale optimization problem with algorithmically given functions. In this chapter, we use sequential and parallel self-configuring genetic algorithm to solve the optimization problem.

The rest of the chapter is organized in the following way. Section 4.2 briefly describes the modeled system. In Sects. 4.3 and 4.4 we describe small size models for two control contours. Section 4.5 illustrates briefly the view of large models. In Sect. 4.6 we describe the proposed optimization algorithm and in Sect. 4.7 we evaluate its performance on the test problems. In Section 4.8, the results of the algorithm performance evaluation on spacecraft control system optimization problems is given, and in the Conclusion section the article content is summarized and future research directions are discussed.

4.2 Problem Description

The system for monitoring and control of an orbital group of telecommunication satellites is an automated, distributed, information-controlling system that includes in its composition on-board control complexes (BCC) of spacecrafts; telemetry, command and ranging (TSR) stations; data telecommunication subsystems; and a mission control center (MCC). The last three subsystems are united in the ground-based control complex (GCC). GCC interacts with BCC(s) through a distributed system of TCR stations and data telecommunication systems that include communication nodes in each TCR, channels and MCC's associated communication equipment. BCC is the controlling subsystem of the spacecraft that ensures real time checking and controlling of on-board systems including pay-load equipment (PLE) as well as fulfilling

program-temporal control. Additionally, BCC ensures the interactivity with ground-based tools of control. The control functions fulfilled by subsystems of the automated control system are considered to form subsets called “control contours” that contain essentially different control tasks. Usually, one can consider the technological control contour, command-programming contour, purpose contour, etc.

Each contour has its own indexes of control quality that cannot be expressed as a function of others. This results in many challenges when attempting to choose an effective control system variant to ensure high control quality with respect to all of the control contours. A multicriterial optimization problem statement is not the only problem. For most of the control contours, criterion cannot be given in the form of an analytical function of its variables but exists in an algorithmic form which requires a computation or simulation model to be run for criterion evaluation at any point.

In order to have the possibility of choosing an effective variant of such a control system, we have to model the work of all control contours and then combine the results in one optimization problem with many models, criteria, constraints and algorithmically given functions of mixed variables. We suggest using evolutionary algorithms (EAs) to solve such optimization problems as these algorithms are known as good optimizers having no difficulties with the described problem properties such as mixed variables and algorithmically given functions. To deal successfully with many criteria and constraints we just have to incorporate techniques, well known in the EA community. However, there is one significant obstacle in the use of EAs for complicated real world problems. The performance of EAs is essentially determined by their settings and parameters which require time and computationally consuming efforts to find the most appropriate ones.

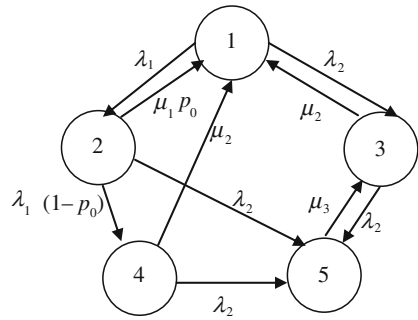
To support the choice of effective variants of spacecrafts’ control systems, we have to develop the necessary models and resolve the problem of EAs settings.

4.3 Technological Control Contour Modelling

The main task of the technological control contour is to provide workability of the spacecraft for the fulfilment of its purposes, i.e., the detecting and locating of possible failures and malfunctions of the control system and pay-load and restoring their lost workability by the activation of corresponding software and hardware tools. The basic index of the quality for this contour is the so called readiness coefficient, i.e., a probability to be ready for work (hasn’t malfunctioned) at each point in time.

We consider simplified control system to describe our modelling approach. Let the system consist of three subsystems: on-board pay-load equipment, on-board control complex and ground-based control complex. Let us assume that GCC subsystems are absolutely reliable but PLE and BCC can fail. If PLE failed, BCC can restore it using its own software tools with the probability p_0 or, otherwise, re-directs restoring process (with the probability $1 - p_0$) into GCC that finishes restoring with the probability equal to one. In the case of a BCC malfunction, GCC restores it with the probability equal to one.

Fig. 4.1 States graph of Markov chain for the simplified model of a technological control contour



We can use a Markov chain approach to model a spacecraft’s control system operation because of its internal features such as high reliability and work stability, e.g., two simultaneous failures are almost impossible, there is no aftereffect if malfunction restoring is finished, etc. That is why we will suppose that all stochastic flows in the system are Poisson ones with corresponding intensities: λ_1 is an intensity of PLE malfunctions, λ_2 is an intensity of BCC malfunctions, μ_1 is an intensity of PLE restoring with BCC, μ_2 is an intensity of PLE restoring with GCC, μ_3 is an intensity of BCC restoring with GCC.

In the described situation, there are five possible states of the system:

1. All subsystems are workable.
2. PLE malfunction, BCC is restoring PLE, GCC is free.
3. BCC malfunction, GCC is restoring BCC, PLE is workable.
4. PLE malfunction, BCC is workable and free, GCC is recovering PLE.
5. PLE malfunction, BCC malfunction, GCC is recovering PLE, and BCC is waiting for recovering.

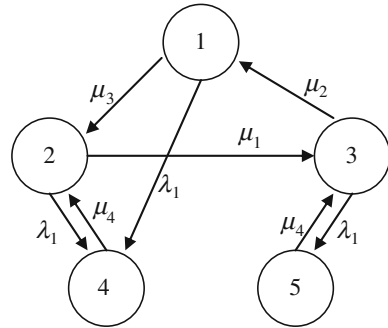
States graph can be drawn as is shown in Fig. 4.1.

Given final probabilities, that the system remains in the corresponding state, as the solution of corresponding Kolmogorov’s equation system, the control quality indicators, i.e., readiness coefficients, can be calculated in following way:

1. Spacecraft readiness coefficient $k_s = P_1$.
2. PLE readiness coefficient $k_{PLE} = P_1 + P_3$.
3. BCC readiness coefficient $k_{BCC} = P_1 + P_2 + P_4$.

To have the effective variant of the spacecraft control system we have to maximize the readiness coefficients subject to constraints on the on-board computer memory and computational efforts needed for the technological contour functions realization. Optimization variables are stochastic flow intensities $\lambda_1, \lambda_2, \mu_1, \mu_2, \mu_3$, as well as p_0 , i.e., the distribution of contour functions between BCC and GCC. If they are characteristics of existing variants of software-hardware equipment, we have the problem of effective variant choice, i.e., a discrete optimization problem. In case of a system preliminary design, some of the intensities can be real numbers and we will have

Fig. 4.2 States graph of Markov chain for simplified model of command-programming control contour



to implement corresponding software and hardware to ensure an optimal solution. Recall that obtained optimization problem has algorithmically given objective functions so before the function value calculation we must solve the equations system.

4.4 Command-Programming Control Contour Modelling

The main task of this contour is the maintenance of the tasks of creating of the command-programming information (CPI), transmitting it to BCC and executing it and control action as well as the realization of the temporal program (TP) regime of control.

We can use Markov chains for modelling this contour for the same reasons. If we suppose that BCC can fail and GCC is absolutely reliable, then we can introduce the following notations: λ_1 is the intensity of BCC failures, μ_1 is the intensity of temporal program computation, μ_2 is the intensity CPI loading into BCC, μ_3 is the intensity of temporal program execution, μ_4 is the intensity of BCC being restored after its failure. The graph of the states for command-programming contour can be drawn as in Fig. 4.2.

There are also five possible states for this contour:

1. BCC fulfills TP, GCC is free.
2. BCC is free, GCC computes TP.
3. BCC is free; GCC computes CPI and loads TP.
4. BCC is restored with GCC which is waiting for continuation of TP computation.
5. BCC is restored with GCC which is waiting for continuation of CPI computation.

BCC, restored after any failure in state one, cannot continue its work and has to wait for a new TP computed with GCC. If BCC has failed in state two or state three then GCC can continue its computation only after BCC restoring completion.

After solving the Kolmogorov's system, we can calculate the necessary indexes of control quality for the command-programming contour. Basic indexes of this contour are the time interval when the temporal program control can be fulfilled without a

change of TP, i.e., the duration of the independent operating of the spacecraft for this contour ($T = P_1/(\mu_2 \cdot P_3)$, has to be maximized); the duration of BCC and GCC interactions when loading TP for the next interval of independent operation of the spacecraft ($t_1 = (P_3 + P_5)/(\mu_1 \cdot P_2)$, has to be minimized); and the average time from the start of TP computation till the start of TP fulfillment by BCC ($t_2 = (P_2 + P_3 + P_4 + P_5)/P_1 \cdot (\lambda_1 + \mu_3)$, has to be minimized).

All these indicators have to be optimized through the appropriate choice of the operations intensities that are the parameters of the software-hardware equipment included in the control system. Corresponding optimization problem has the same properties as described above.

4.5 Models Generalization

We described above the simplified models of two control contours in order to demonstrate the modelling technique. The developed models are not adequate for the use in the spacecraft control system design process because of the unrealistic assumption of GCC reliability.

If we suppose the GCC can fail then we have to add the states when GCC fails while the system is in any state.

GCC in our problem description consists of three subsystems groups—TCR stations, data communication subsystem, and MCC equipment. Considering all three groups as one unit, we will have three GCC subsystems. If any of them can fail, the new nodes and possible transitions have to be added into model.

We will not describe the meaning of all notion in details, recall that λ_i indicate the intensities of subsystems failures and μ_j indicate the intensities of subsystems being restoring by BCC (for PLE) or GCC (for all subsystems including itself).

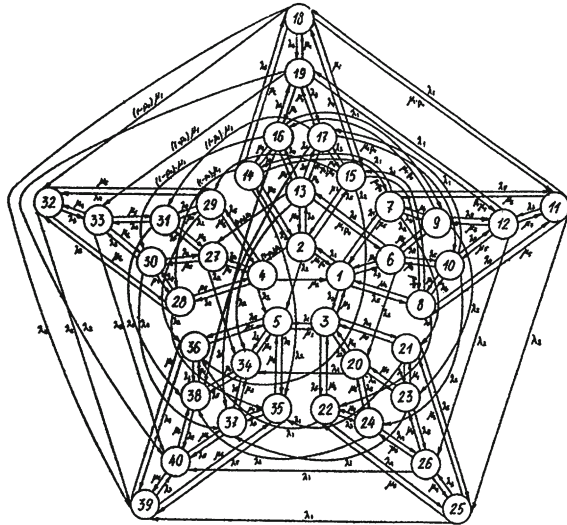
The states graph for this case consists of 40 states and 146 transitions and is depicted in Fig. 4.3. Corresponding Kolmogorov's equation system contains 40 lines.

Under the same conditions, the states graph for the command-programming contour consists of 96 states and more than 300 transitions and cannot be shown here.

Going deeper into the details we must continue dividing the subsystems groups (TCR stations, telecommunication system, MCC) in parts. Then we must unify models of all contours of the spacecraft and models of identical contours of different spacecrafts of the orbital group. Additionally, in some cases we cannot use simple Markov chain models and need a more sophisticated simulation models. Certainly, this work cannot be done without an adequate computation tool.

In the next stage of our research we have developed and implemented a decision support system for spacecraft control systems modelling with stochastic processes models. This DSS suggests questions to aerospace engineers designing spacecraft control systems in their terms relative to system structure, its subsystems, possible states and transitions, executed operations, etc. Giving the answers to these questions the DSS generates the necessary data structure, the lists of states and transitions with

Fig. 4.3 States graph of Markov chain for modelling technological control contour with unreliable GCC subsystems



their descriptions in designer terms and definitions, Kolmogorov’s equations system, etc. [9].

This DSS is able also to solve optimization problems with some adaptive search algorithms. As has been stressed above, optimization problems arising in the described situation are hard to solve. That is why we suggest here using our modified genetic algorithm.

4.6 Optimization Algorithms Description

Evolutionary algorithms (EA), the best known representatives of which are genetic algorithms (GA), are well known optimization techniques based on the principles of natural evolution [2]. Although GAs are successful in solving many real world optimization problems [6], their performance depends on the selection of the GA settings and tuning their parameters [3]. GAs usually use a bit-string solution representation, but other decisions have to be made before the algorithm can run. The design of a GA consists of the choice of variation operators (e.g. recombination and mutation) that will be used to generate new solutions from the current population and the parent selection operator (to decide which members of the population are to be used as inputs to the variation operators), as well as a survival scheme (to decide how the next generation is to be created from the current one and outputs of the variation operators). Additionally, real valued parameters of the chosen settings (the probability of recombination, the level of mutation, etc.) have to be tuned [3].

The process of settings determination and parameters tuning is known to be a time-consuming and complicated task. There exist much research devoted to “self-adapted” or “self-tuned” GA and authors of the corresponding works determine

similar ideas in very different ways, all of them aimed at reducing the role of human expert in algorithms designing.

The main idea of the approach used in this chapter relies to automated selecting and using existing algorithmic components. That is why our algorithms might be called as self-configuring ones.

In order to specify our algorithms more precisely, one can say that we use dynamic adaptation on the level of population. The probabilities of applying the genetic operators are changed “on the fly” through the algorithm execution. According to the classification given in [5] we use centralized control techniques (central learning rule) for parameter settings with some differences from the usual approaches. Operator rates (the probability to be chosen for generating off-spring) are adapted according to the relative success of the operator during the last generation independently of the previous results. This is why our algorithm avoids problem of high memory consumption typical for centralized control techniques [5]. Operator rates are not included in individual chromosome and they are not subject to the evolutionary process. All operators can be used during one generation for producing off-spring one by one.

Having in mind the necessity to solve hard optimization problems and our intention to organize GA self-adaptation to these problems, we must first improve the GA flexibility before it can be adapted. For this reason we have tried to modify the most important GA operator, i.e., crossover.

The uniform crossover operator is known as one of the most effective crossover operators in conventional genetic algorithm [1, 11]. Moreover, nearly the beginning, it was suggested to use a parameterized uniform crossover operator and it was shown that tuning this parameter (the probability for a parental gene to be included in off-spring chromosome) one can essentially improve “The Virtues” of this operator [1]. Nevertheless, in the majority of cases using the uniform crossover operator the mentioned possibility is not adopted and the probability for a parental gene to be included in off-spring chromosome is given equal to 0.5 [2, 6].

Thus it seems interesting to modify the uniform crossover operator with an intention to improve its performance. Desiring to avoid real number parameter tuning, we suggested introducing selective pressure on the stage of recombination [8] making the probability of a parental gene to be taken for off-spring dependable on parent fitness values. Like the usual GA selection operators, fitness proportional, rank-based and tournament-based uniform crossover operators have been added to the conventional operator called here the equiprobable uniform crossover.

Although the proposed new operators, hopefully, give higher performance than the conventional operators, at the same time the number of algorithm setting variants increases that complicates algorithms adjusting for the end user. That is why we need suggesting a way to avoid this extra effort for the adjustment.

With this aim, we apply operators’ probabilistic rates dynamic adaptation on the level of population with centralized control techniques. To avoid real parameters precise tuning, we use setting variants, namely types of selection, crossover, population control and a level of mutation (medium, low, high). Each of these has its own probability distribution. E.g., there are five settings of selection—fitness pro-

portional, rank-based, and tournament-based with three tournament sizes. During the initialization all probabilities are equal to 0.2 and they will be changed according to a special rule through the algorithm's execution in such a way that a sum of probabilities should be equal to one and no probability could be less than a pre-conditioned minimum balance. The list of crossover operators includes 11 items, i.e., one-point, two-point and four uniform crossovers all with two numbers of parents (two and seven).

When the algorithm has to create the next off-spring from the current population, it firstly has to configure settings, i.e. to form the list of operators with the use of the probability operator distributions. Then the algorithm selects parents with the chosen selection operator, produces an off-spring with the chosen crossover operator, mutates this off-spring with the chosen mutation probability and puts it into the intermediate population. When the intermediate population is filled, the fitness evaluation is executed and operator rates (the probabilities to be chosen) are updated according to the operator productivity. Then the next parental population is formed with the chosen survival selection operator. The algorithm stops after a given number of generations or if another termination criterion is met.

The productivity of an operator is the ratio of the average off-springs fitness obtained with this operator and the off-spring population average fitness. The successful operator, having maximal productivity, increases its rate obtaining portions from other operators. There is no necessity for extra computer memory to remember past events and the reaction of updates are more dynamic.

The second problem of GAs application in the real world problem solving is their execution time. GAs need enough members of population and evolution generations to demonstrate high performance. However, the fitness function evaluation in real world problems is usually time-consuming, e.g. in our case it is necessary to solve equations system with many tens of equations. We tackle this problem through the Self CGA parallelization in two ways: simple distribution of individuals' fitness evaluation among many computing cores of usual multicore computer and the application of island-based GA framework [7].

The first approach admits the SelfCGA speed up proportionally to the number of cores without advantages in performance (solution quality and reliability) and serves mainly as a reference base for the second approach. The second approach is the eight cores based island SelfCGA with clique connection graph between core populations which allow migration of the 5% of the best individuals after every 10% of the whole amount of generations number. Every population on the computing core evolves independently between migration time points. It is expected that this approaches will give not only algorithm speed up but also a positive effect on the performance.

Table 4.1 Comparison results of SelfCGA and problem single best algorithms

No	Crossover	Average	Min	Max	Average	Min	Max
1	UE	0.818	0.787	0.894	0.838	0.807	0.904
	SelfCGA	<i>0.886</i>			0.902		
2	UE	0.841	0.808	0.903	0.849	0.851	0.909
	SelfCGA	<i>0.866</i>			0.901		
3	UE	0.901	0.887	0.921	0.911	0.898	0.951
	SelfCGA	<i>0.901</i>			0.946		
4	UR	0.925	0.877	0.959	0.941	0.897	0.979
	SelfCGA	0.976			0.991		
5	UT	0.950	0.901	1.00	0.972	0.911	1.000
	SelfCGA	1.000			1.000		
6	UE	0.953	0.861	0.999	0.981	0.893	1.000
	SelfCGA	1.00			1.000		
7	UT	0.897	0.832	0.927	0.911	0.876	0.957
	SelfCGA	0.878			0.920		
8	UR	0.741	0.667	0.800	0.790	0.693	0.830
	SelfCGA	0.830			0.935		
9	UT	0.967	0.917	0.983	0.977	0.931	0.990
	SelfCGA	0.987			1.000		
10	UE	0.853	0.803	0.891	0.865	0.824	0.909
	SelfCGA	<i>0.884</i>			0.897		
11	UR	0.821	0.734	0.888	0.867	0.793	0.955
	SelfCGA	0.892			0.995		
12	UR	0.833	0.765	0.881	0.834	0.776	0.898
	SelfCGA	0.897			0.995		
13	UR	0.956	0.902	0.998	0.976	0.920	1.000
	SelfCGA	1.000			1.000		
14	UR	0.974	0.935	0.999	0.989	0.985	1.000
	SelfCGA	1.000			1.000		

4.7 Algorithms Performance Evaluation

The performance of a conventional GA with three additional uniform crossover operators has been evaluated on the 14 usual test problems for GA [4]. After 1,000 runs and statistical processing of the results, the following observations were found in terms of algorithm reliability [10]. The best variants are the new rank-based and conventional (equiprobable) uniform operators. Tournament-based crossover seems to be weak but it is the only operator having maximum reliability of 100% on some test problems where other operators fail. The reliability is the percentage of the algorithm's runs that give satisfactorily precise solutions.

The next stage in evaluating the algorithms is a comparison with the proposed self-configuring GA (SelfCGA). In Table 4.1 one can find the results comparing SelfCGA with the single best algorithm having had the best performance on the corresponding problem. In Table 4.1, headers "UE, UT, UP, UR" indicate the type of

crossover, respectively, uniform equiprobable, uniform tournament-based, uniform fitness proportional and uniform rank-based crossovers.

Saying “single” algorithm, we mean the group of algorithms with the same crossover operator but with all variants of other settings. The average reliability of this “single” algorithm is averaged over all possible settings. “Min” and “Max” mean GA settings given the worst and the best performance on the corresponding test problem.

Analyzing the first three columns of Table 4.1, related to sequential SelfCGA, we can see that in four cases (1, 2, 3, 10 numbers are given in italics) SelfCGA demonstrates better reliability than the average reliability of the corresponding single best algorithm but worse than the maximal one. In one case (7th problem), the single best algorithm (with tournament-based uniform crossover) gives better average performance than SelfCGA. In the remaining nine cases (numbers are given in bold) SelfCGA outperforms even the maximal reliability of the single best algorithm. Similar results we can observe in the last three columns related to parallel (islands based) SelfCGA. Additionally, we can observe that parallel SelfCGA demonstrates higher reliability than sequential one in all cases.

Having described these results, one can conclude that the proposed way of GA self-configuration not only eliminates the time consuming effort for determining the best settings but also can give a performance improvement even in comparison with the best known settings of conventional GA. It means that we may use the SelfCGA in real world problems solving.

4.8 Self-Configuring Genetic Algorithm Application in Spacecraft Control System Design

First of all we evaluate its performance on the simplified models of technological and command-programming control contours with five states.

To choose an effective variant of the technological control contour we have to optimize the algorithmically given function with six discrete variables. The optimization space contains about 1.67×10^7 variants and can be enumerated with an exhaustive search within a reasonable time. In such a situation, we know the best (k^*) and the worst (k^-) admissible values of indicators. Executing 100 runs of the algorithm, we will also know the worst value of indicators (k^*) obtained as a run result. The best result of the run should be (k^*) if the algorithm finds it. We use 20 individuals in one generation and 30 generations for one run for all eight cores. This means the algorithm will examine 4,800 points of the optimization space, i.e. about 0.029 % of it. These points are distributed in equal parts between eight cores for parallel SelfCGA. As the indicators of the algorithm performance we will use the reliability (the percentage of the algorithm’s runs that give the exact solution k^*); maximum deviation MD (the ratio of k^*-k^* and k^* in percentage to the last); and relative maximum deviation RMD (the ratio of k^*-k^* and k^*-k^- in percentage to the last). The comparison is made for five algorithms, namely four conventional GAs

Table 4.2 Algorithm reliability comparison for technological control contour model with five states (spacecraft readiness coefficient)

Algorithm	Reliability 1	MD (%)	RMD (%)	Reliability 8	MD (%)	RMD (%)
UE	0.91	0.0020	0.3560	0.93	0.0011	0.3137
UR	0.94	0.0015	0.2805	0.96	0.0012	0.2349
UP	0.86	0.0021	0.3807	0.91	0.0011	0.3287
UT	0.98	0.0007	0.1243	0.98	0.0004	0.1122
SelfCGA	1.00	0.0000	0.0000	1.00	0.0000	0.0000

Table 4.3 Algorithm reliability comparison for command-programming control contour model with five states

Algorithm	Ind.	Reliability 1	MD (%)	RMD (%)	Reliability 8	MD (%)	RMD (%)
UE	T	0.87	6.431	9.028	0.91	5.243	7.820
	t ₁	0.76	0.956	3.528	0.81	0.764	2.852
	t ₂	0.83	13.392	26.010	0.87	12.312	24.212
UR	T	0.95	3.987	5.600	0.97	3.272	5.010
	t ₁	0.93	0.341	1.258	0.96	0.276	1.004
	t ₂	0.93	11.347	22.040	0.95	11.021	20.64
UP	T	0.79	6.667	9.359	0.87	5.468	8.561
	t ₁	0.71	1.156	4.266	0.78	1.007	3.876
	t ₂	0.74	16.321	31.700	0.81	14.241	27.700
UT	T	0.91	4.873	6.840	0.94	4.012	5.920
	t ₁	0.81	0.956	3.528	0.88	0.761	2.988
	t ₂	0.86	13.392	26.010	0.90	12.091	22.11
SelfCGA	T	1.00	0.000	0.000	1.00	0.000	0.000
	t ₁	0.98	0.092	0.215	1.00	0.000	0.000
	t ₂	1.00	0.000	0.000	1.00	0.000	0.000

with new uniform crossover operators (UE, UR, UP, UT) and SelfCGA. For conventional GAs, the results are given for the best determination of all other settings. In Table 4.2, the results are shown for single SelfCGA (one core) together with the island model parallel SelfCGA (eight cores).

Similar evaluations for all three indicators of the command-programming contour are given in Table 4.3.

The difference exists in the optimization problem size. 4,800 evaluations of the objective function correspond to 0.456% of the whole optimization space as the model has only five discrete variables (about 10^6 variants).

From Tables 4.2 and 4.3 one can see that SelfCGA outperforms the alternative algorithms for all problem statements and with all performance measures. Additionally, parallel algorithms outperform sequential ones having the same number of fitness evaluations.

Now we have to evaluate the performance of the suggested algorithm on generalized models which have much higher dimensions.

Table 4.4 Algorithm reliability comparison for technological control contour model with 40 states (spacecraft readiness coefficient)

Algorithm	Reliability 1	MD (%)	RMD (%)	Reliability 8	MD (%)	RMD (%)
UE	0.83	0.0105	0.2008	0.88	0.0099	0.1787
UR	0.89	0.0089	0.1750	0.92	0.0078	0.1565
UP	0.78	0.0101	0.2102	0.85	0.0093	0.2001
UT	0.90	0.0075	0.1592	0.93	0.0071	0.1471
SelfCGA	0.96	0.0081	0.1442	0.99	0.0009	0.0121

Table 4.5 Algorithm reliability comparison for command-programming control contour model with 96 states

Algorithm	Indicator	Reliability 1	Reliability 8
UE	T	0.76	0.78
	t ₁	0.67	0.71
	t ₂	0.75	0.78
UR	T	0.84	0.85
	t ₁	0.81	0.84
	t ₂	0.84	0.87
UP	T	0.70	0.76
	t ₁	0.59	0.65
	t ₂	0.63	0.67
UT	T	0.83	0.89
	t ₁	0.72	0.77
	t ₂	0.77	0.80
SelfCGA	T	0.91	0.98
	t ₁	0.87	0.95
	t ₂	0.89	0.96

The optimization model for the technological control contour has 11 discrete variables. The corresponding optimization space contains about 1.76×10^{13} points and cannot be enumerated with an exhaustive search especially if one recalls that the examination of one point includes solving a linear equations system with 40 variables. The best (k^*) and the worst (k^-) admissible values of indicators cannot be given and we use here their best known evaluations after multiple runs consuming much computational resources. Nevertheless, we still can try to obtain the resulting table similar to Table 4.2 with statistical confidence. For the algorithms performance evaluations we use 40 individuals for one generation and 80 generation for one run on the one core that examines about $1.456 \times 10^{-6} \%$ of the search space examination (sequential algorithms have 100 individuals and 256 generation that give same 25600 fitness evaluations). Results of numerical experiments for one core and eight cores are summarized in Table 4.4.

For the last problem, i.e. for the model of the command-programming control contour with 96 states and more than 300 transitions, we cannot give detailed information as we did above. This problem has 13 variables and contains 4.5×10^{15} points in the optimization space. It requires enormous computational efforts to find reliable

evaluations of the necessary indicators. Instead, we give a smaller table without MD and RMD measures. It gives us some insight on the comparative reliability of the investigated algorithms. The algorithms performance evaluation requires the examination of 1.76×10^{-9} % of the search space (100 individuals and 100 generations for one computing core in parallel SelfCGA and 320 individuals and 250 generations for sequential variant). Results averaged over 100 runs are summarized in Table 4.5.

Tables 4.4 and 4.5 show that the SelfCGA outperforms all alternative algorithms and its parallel variant outperforms sequential one.

We have no place to go into the details with estimations of parallel algorithms speed up. Nevertheless, we can remark that simple parallelization of SelfCGA on the eight core computing system accelerates the execution in almost eight times. It can be explained if we realise that any fitness evaluation is time consuming process comparing with algorithm operations. We have also to remark that island model SelfCGA additionally accelerates execution because of the synergetic effect of the cooperation of independently evolving populations. This effect gives also positive impact on the algorithm performance.

4.9 Conclusions

In this chapter, the mathematical models in the form of Markov chains have been developed and implemented for choosing effective variants of spacecraft control contours. These models contain tens of states and hundreds of transitions that make the corresponding optimization problems hard to solve.

It is suggested to use the genetic algorithms in such a situation because of their reliability and high potential to be problem adaptable. As GAs performance is highly dependent on their setting determination and parameter tuning, the special self-configuring GA is suggested that eliminates this problem. The high performance of the suggested algorithm is demonstrated through experiments with test problems and then is validated by the solving hard optimization problems. The self-configuring genetic algorithm is suggested to be used for choosing effective variants of spacecraft control systems as it is very reliable and requires no expert knowledge in evolutionary optimization from end users (aerospace engineers). We did not try to implement the best known GA with optimal configuration and optimally tuned parameters. Certainly, one could easily imagine that the much better GA exists. However, it is a problem to find it for every problem in hand. The way of the self-configuration proposed in this chapter that involves all variants of all operators can be easily expanded by adding new operators or operator variants. The self-configuring process monitoring gives the additional information for further SelfCGA improving. E.g., if the high level mutation is always the winner among mutation variants then we can add some higher level mutation operators in the competitors list instead of lower level variants.

The future research includes also not only direct expansion in using the simulation models and multicriterial optimization problem statements but also the improvement

of SelfCGA adaptability through the population size control and adoption of additional operators and operator variants as well as effective variants of parallelization.

Acknowledgments The research is supported through the Governmental contracts No 16.740.11.07 42 and No 11.519.11.4002. The authors are deeply grateful to Dr. Linda Ott, a professor at the Technological University of Michigan, for her invaluable help in improving the text of the article.

References

1. De Jong, K. A., Spears, W.: On the virtues of parameterized uniform crossover. In: Belew, R. K., Booker, L. B. (eds.) *Proceedings of the 4th International Conference on Genetic Algorithms*, pp. 230–236. Morgan Kaufmann (1991)
2. Eiben, A.E., Smith, J.E.: *Introduction to Evolutionary Computing*. Springer-Verlag, Berlin, Heidelberg (2003)
3. Eiben, A. E., Hinterding, R., Michalewicz, Z.: Parameter control in evolutionary algorithms. *IEEE Trans. Evol. Comput* **3**(2), 124–141 (1999) (IEEE press piscataway, NJ)
4. Finck, S., Hansen, N., Ros, R., Auger, A.: Real-parameter black-box optimization benchmarking 2009: presentation of the noiseless functions. Technical Report 2009/20, Research center PPE (2009)
5. Gomez, J.: Self-adaptation of operator rates in evolutionary algorithms. In: Deb, K., et al. (eds.) *GECCO 2004, LNCS 3102*, pp. 1162–1173. Springer, Heidelberg (2004)
6. Haupt, R.L., Haupt, S.E.: *Practical Genetic Algorithms*. Wiley, Hoboken (2004)
7. Lardeux, F., Goëffon, A.: A dynamic island-based genetic algorithms framework. In: *Simulated evolution and learning, lecture notes in computer science 6457*, pp. 156–165. Springer, Heidelberg (2010)
8. Semenkin, E. S., Semenkina, M. E.: Application of genetic algorithm with modified uniform recombination operator for automated implementation of intellectual information technologies. In: *Vestnik. Scientific Journal of the Siberian State Aerospace University named after academician M. F. Reshetnev. - Issue 3 (16)*, 27–32. (In Russian, abstract in English). SibSAU press, Krasnoyarsk (2007)
9. Semenkin, E., Semenkina, M.: Spacecrafts' control systems effective variants choice with self-configuring genetic algorithm. In: Ferrier, J.-L., Bernard, A., Gusikhin, O., Madani, K. (eds), *Proceedings of the 9th International Conference on Informatics in Control, Automation and Robotics*, vol. 1, pp. 84–93 (2012)
10. Semenkin, E., Semenkina, M.: Self-configuring genetic algorithm with modified uniform crossover operator. In: Tan, Y., Shi, Y., Ji, Z. (Eds.): *Advances in Swarm Intelligence, ICSI 2012, Part I, LNCS 7331*, pp. 414–421. Springer, Heidelberg (2012)
11. Syswerda, G.: Uniform crossover in genetic algorithms. In: Schaffer, J. (ed.) *Proceedings of the 3rd International Conference on Genetic Algorithms*, pp. 2–9. Morgan Kaufmann, San Francisco (1989)

Chapter 5

A Heuristic Control Algorithm for Robust Internal Model Control with Arbitrary Reference Model

M. G. Skarpetis, F. N. Koumboulis and A. S. Ntellis

Abstract In this chapter the problem of Robust Internal Model Control is considered for the case of linear plants with nonlinear uncertain structure. The reference command is produced by an arbitrary reference model. A finite step Heuristic Algorithm is proposed in order to derive the controller parameters that guarantee robust performance under the proposed solvability conditions. The proposed controller is successfully applied to a hydraulic actuator uncertain model including uncertain parameters arising from changes of the operating conditions and other physical reasons. The satisfactory performance of hydraulic actuator variables for all the expected range of the actuator model uncertainties and external disturbances is illustrated via simulation experiments.

Keywords Robust internal model control · Arbitrary reference command tracking · Hydraulic actuator

5.1 Introduction

The problem of output tracking appears to be one of the most popular control design problems (see [1–5] and the reference therein). The problem of output tracking for both non uncertain and uncertain systems (robust tracking) is treated mainly using stabilizability techniques, e.g. [3, 5–7].

M. G. Skarpetis (✉) · F. N. Koumboulis · A. S. Ntellis
Department of Automation, Halkis Institute of Technology, Psahna Evoias,
34400 Halkis, Greece
e-mail: skarpetis@teihal.gr

F. N. Koumboulis
e-mail: koumboulis@teihal.gr

A. S. Ntellis
e-mail: ntellis@teihal.gr

The problem of robust tracking appears to be of major interest in the design of controllers for hydraulic actuators. This type of actuators is widely used in many applications like manufacturing, robotics, constructions and avionics. The dynamics of fluid power are inherently uncertain. So, robust control strategies are indispensable if one wishes to guarantee safety and reliability of hydraulic actuators (see [8–12] and the references therein). Robust asymptotic tracking techniques like those in [6–8, 12] and robust PI-PID design techniques like those in [10, 11] and the reference therein, perform satisfactory in many industrial hydraulic plants.

In this chapter a robust tracking controller is proposed in order to satisfy asymptotic command following for reference signals produced by an arbitrary reference model. The design technique is based on the well known Internal Model Principle [4], appropriately extended using Hurwitz invariability in order to cover the case of linear uncertain systems with non linear uncertain structure. An arbitrary reference model that produces desired reference signals is used in the controller structure and the overall closed loop robust stability is guaranteed under sufficient conditions. A finite step Heuristic Algorithm is proposed in order to derive the controller parameters solving the problem.

The present results are successfully applied to control the position of a hydraulic actuator model involving uncertain parameters arising from changes of the operating conditions (temperature, pressure, entrained air or water) as well as physical uncertainties (loss in the effective area of the actuator piston seal due to wear [9]). Solvability conditions are established. An analytic finite step algorithm for the computation of the robust controller parameters is proposed. Following the algorithm, first, robust stability regions are determined. Second, the metaheuristic optimization algorithm proposed in [13] is applied, inside these regions, to fulfill performance criteria. The effectiveness of the controller is illustrated through simulations for various values of the model uncertain parameters. The results appear to be simple and easily applicable. The present chapter is an extended and upgraded version of [14].

5.2 Preliminary Results

Consider the linear time-invariant SISO system with non linear uncertain structure described by

$$\dot{x}(t) = A(q)x(t) + b(q)u(t) + d(q)w(t), \quad y(t) = c(q)x(t) \quad (5.1)$$

where $x(t) \in \mathbb{R}^n$ is the state vector, $u(t) \in \mathbb{R}$ is the input and $y(t) \in \mathbb{R}$ is the output and $w(t) \in \mathbb{R}$ is external disturbance. $A(q) \in [\wp(q)]^{n \times n}$, $b(q) \in [\wp(q)]^{n \times 1}$, $d(q) \in [\wp(q)]^{n \times 1}$ and $c(q) \in [\wp(q)]^{1 \times n}$ are function matrices depending upon the uncertainty vector $q = [q_1 \cdots q_l] \in \mathbb{Q}$ (\mathbb{Q} denotes the uncertain domain). The set $\wp(q)$ is the set of nonlinear functions of q . The uncertainties q_1, \dots, q_l do not depend upon the time. With regard to the nonlinear structure of $A(q)$, $b(q)$, $d(q)$ and $c(q)$ no limitation or specification is considered (i.e. boundness, continuity).

Consider the case where the reference signal $y_r(t)$ is the output of a linear model described by

$$\dot{x}_r(t) = A_r x_r(t); \quad y_r(t) = c_r x_r(t) \quad (5.2)$$

where $y_r(t) \in \mathbb{R}$, $x_r(t) \in \mathbb{R}^{r \times 1}$ and where

$$A_r = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ -d_r & -d_{r-1} & -d_{r-2} & \cdots & -d_1 \end{bmatrix}, \quad c_r = [1 \ 0 \ \cdots \ 0].$$

Consider the vector $x_{r,0}$ denoting arbitrary initial conditions for system (5.2). Clearly, it holds that

$$y_r^{(r)}(t) + \sum_{i=1}^r d_i y_r^{(r-i)}(t) = 0. \quad (5.3)$$

The disturbance is assumed to be of the same type as the reference signal, i.e.

$$w^{(r)}(t) + \sum_{i=1}^r d_i w^{(r-i)}(t) = 0. \quad (5.4)$$

Define the tracking error

$$\varepsilon(t) = y(t) - y_r(t). \quad (5.5)$$

Differentiating the error r-times, we get

$$\varepsilon^{(r)}(t) + \sum_{i=1}^r d_i \varepsilon^{(r-i)}(t) = c(q)x^{(r)}(t) + c(q) \sum_{i=1}^r d_i x^{(r-i)}(t). \quad (5.6)$$

Define the variables

$$z(t) = x^{(r)}(t) + \sum_{i=1}^r d_i x^{(r-i)}(t), \quad \tilde{u}(t) = u^{(r)}(t) + \sum_{i=1}^r d_i u^{(r-i)}(t). \quad (5.7)$$

According to (5.3), (5.4), (5.6) and (5.7) the following augmented system is defined:

$$\frac{d}{dt} \tilde{x}(t) = \tilde{A}(q) \tilde{x} + \tilde{b}(q) \tilde{u}(t) \quad (5.8)$$

where $\tilde{x}(t) = [\varepsilon(t) \ \varepsilon^{(1)}(t) \ \dots \ \varepsilon^{(r-1)}(t) \ z(t)]^T$, $\tilde{A}(q) = \left[\begin{array}{c|c} A_r & e_r c(q) \\ \hline - & - \end{array} \right]$,
 $\tilde{b}(q) = \left[\begin{array}{c} 0_{r \times 1} \\ b(q) \end{array} \right]$ and $e_r = \left[\begin{array}{c} 0_{(r-1) \times 1} \\ 1 \end{array} \right]$.

Consider the static state feedback control law

$$\tilde{u}(t) = f\tilde{x}(t) = f_1\tilde{\varepsilon}(t) + f_2z(t). \quad (5.9)$$

where $\tilde{\varepsilon}(t) = [\varepsilon(t) \ \varepsilon^{(1)}(t) \ \dots \ \varepsilon^{(r-1)}(t)]^T$.

According to the aforementioned definitions the robust output command tracking is formulated as follows [1, 4]: The output of the uncertain system (5.1) follows the output of the reference system (5.2) while the tracking error (5.5) decreases asymptotically to zero. This is satisfied using a static state feedback control law of the form (5.9) guaranteeing robust stability of the polynomial

$$\tilde{p}_{cl}(s, q, f) = \det [sI_{r+n} - \tilde{A}(q) - \tilde{b}(q)f]. \quad (5.10)$$

The control law (5.9) can be expressed in state space form as follows:

$$\dot{x}_c(t) = A_c x_c(t) + b_c \varepsilon(t), \quad v(t) = c_c x_c(t), \quad u(t) = v(t) + f_2 x(t) \quad (5.11a)$$

$$A_c = \begin{bmatrix} -d_1 & 1 & 0 & \dots & 0 \\ -d_2 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ -d_r & 0 & 0 & \dots & 0 \end{bmatrix}, \quad b_c = \begin{bmatrix} f_{1,r} \\ f_{1,r-1} \\ \vdots \\ f_{1,1} \end{bmatrix}, \quad c_c = [1 \ 0 \ \dots \ 0]. \quad (5.11b)$$

where $f_{1,i} (i = 1, \dots, r)$ are the elements of f_1 .

5.3 Solvability Conditions

The polynomial (5.10) can be rewritten as

$$\tilde{p}_{cl}(s, q, f) = \det [sI_r - A_r] \det [sI_n - A(q)] - f \operatorname{adj} [sI_{n+r} - \tilde{A}(q)] \tilde{b}(q). \quad (5.12)$$

Define:

$$\tilde{a}(q) = [1 \ \tilde{a}_1(q) \ \dots \ \tilde{a}_{r+n}(q)] \quad (5.13)$$

where $\tilde{a}_i(q) (i = 1, \dots, r+n)$ are the coefficients of the polynomial $\det [sI_r - A_r] \det [sI_n - A(q)]$. Also define the polynomial matrix

$$\tilde{P}(s, q) = \Omega(q) [s^{\mu(q)} \ \dots \ s^0]^T = \operatorname{adj} [sI_{n+r} - \tilde{A}(q)] \tilde{b}(q)$$

where $\mu(q) \leq n + r$ is the maximum degree of the polynomial matrix $\text{adj} [sI_{n+r} - \tilde{A}(q)] \tilde{b}(q)$ and

$$\Omega(q) = [\omega_0(q) \dots \omega_{\mu(q)}(q)]; \quad \omega_i(q) = [\tilde{\omega}_{i,1}(q) \dots \tilde{\omega}_{i,n+r}(q)]^T. \quad (5.14)$$

According to definitions (5.13) and (5.14) the augmented closed loop characteristic polynomial (5.12) can equivalently be expressed as follows:

$$\tilde{p}_{cl}(s, q, f) = [s^{n+r} \dots s^0] A^{**}(q) \begin{bmatrix} 1 \\ f^T \end{bmatrix} \quad (5.15a)$$

$$A^{**}(q) = [\tilde{a}^T \quad -\tilde{\Omega}^T], \quad \tilde{\Omega}(q) = [0_{(n+r) \times (n+r-\mu(q))} \mid \Omega(q)] \quad (5.15b)$$

Based on the above definitions and the results in [15–17] the following theorem is presented.

Theorem 1 The problem of robust output command tracking for the uncertain system (5.1) and for arbitrary signals produced by the reference model (5.2), is solvable, via the controller (5.9), if the following conditions are satisfied

- (i) The elements of $A^{**}(q)$ are continuous functions of q for every $q \in \mathbb{Q}$
- (ii) There exists $(n + r + 1)$ —row submatrix of $A^{**}(q)$, let $A^*(q)$ which is positive antisymmetric.

Proof According to the definition of the problem presented in Sect. 5.2, the problem of robust output command tracking for the uncertain system (5.1) via the controller (5.9) is solvable if the polynomial (5.10) or equivalently (5.12) is robustly stable. According to the results in [15–17] the uncertain polynomial is Hurwitz invariant if conditions (i) and (ii) of Theorem 1 are satisfied.

In the following theorem necessity is studied.

Theorem 2 For the problem of robust output command tracking for the uncertain system (5.1) and for arbitrary signals produced by the reference model (5.2), via the controller (5.9), it is necessary for the roots of the polynomial $c(q)\text{adj} [sI_n - A(q)] b(q)$ not to be unstable roots of $\det [sI_r - A_r]$ for every $q \in \mathbb{Q}$.

Proof The polynomial (5.12) can be rewritten as follows

$$\begin{aligned} \tilde{p}_{cl}(s, q, f) = & \det [sI_r - A_r] \det [sI_n - A(q)] \\ & + f_1 \text{adj} [sI_r - A_r] c(q) \text{adj} [sI_n - A(q)] b(q) - \\ & + f_2 \text{adj} [sI_n - A(q)] b(q) \det [sI_r - A_r] \end{aligned}$$

From the above relation it is clear that if $\sigma(q) \in \mathbb{C}$ is a root of $c(q)\text{adj}[sI_n - A(q)]b(q)$, being unstable for at least one $q \in \mathbb{Q}$. Then its value for this q must not be eigenvalue of $\det[sI_r - A_r]$.

The condition of Theorem 2 is related to the uncontrollable part of the augmented system $(\tilde{A}(q), \tilde{b}(q))$. This condition is useful in choosing the model of the reference signal.

Remark 1 The class of the systems that satisfy condition (ii) of Theorem 1, can be widen, if, instead of $A^{**}(q)$ the matrix $A^{**}(q)T$ is considered where T is an appropriate invertible and independent from q matrix.

For the definition of positive antisymmetric matrices see [15–17].

5.4 An Analytic Algorithm for the Computation of the Controller Parameters

According to [15–17] and under the satisfaction of the solvability conditions the Hurwitz invariability matrix $A^*(q)$ can be constructed using $(n+r+1) - \xi$ up or down augmentations starting from a positive Hurwitz invariant core with ξ rows as follows: $\Phi_1(q) \rightarrow \Phi_2(q) \rightarrow \dots \rightarrow \Phi_{n+r+1-\xi}(q) \rightarrow \Phi_{n+r+2-\xi}(q) = A^*(q)$.

Based on the above construction the following heuristic algorithm (Fig. 5.1) based on the respective results in [13, 15–17] is presented using indicatively up augmentations for the construction of the stability augmentation matrix:

Step 1 (Construction of the augmentation matrices): The core of $A^*(q)$ is $\bar{c}(q) = \Phi_1(q)$. From $\Phi_1(q)$ using $n+r+1-\xi$ up positive augmentations the matrices $\Phi_2(q), \dots, \Phi_{n+r+2-\xi}(q) = A^*(q)$ are constructed.

Step 2 (Initialization): Define $\tau_1 = 1, i = 1$.

Step 3 (Determination of the region of $\varepsilon_i > 0$ for which $\Phi_{i+1}(q) [\varepsilon_i \tau_i]^T$ is positive Hurwitz invariant): According to the form of the associated polynomial of $\Phi_{i+1}(q) [\varepsilon_i \tau_i]^T$, find a region of ε_i , let M_i , where in the robust stability is guaranteed under the constrains of the stability regions M_1, \dots, M_{i-1} . Let $\tau_{i+1} = [\varepsilon_i \ 1]$, and $i = i + 1$.

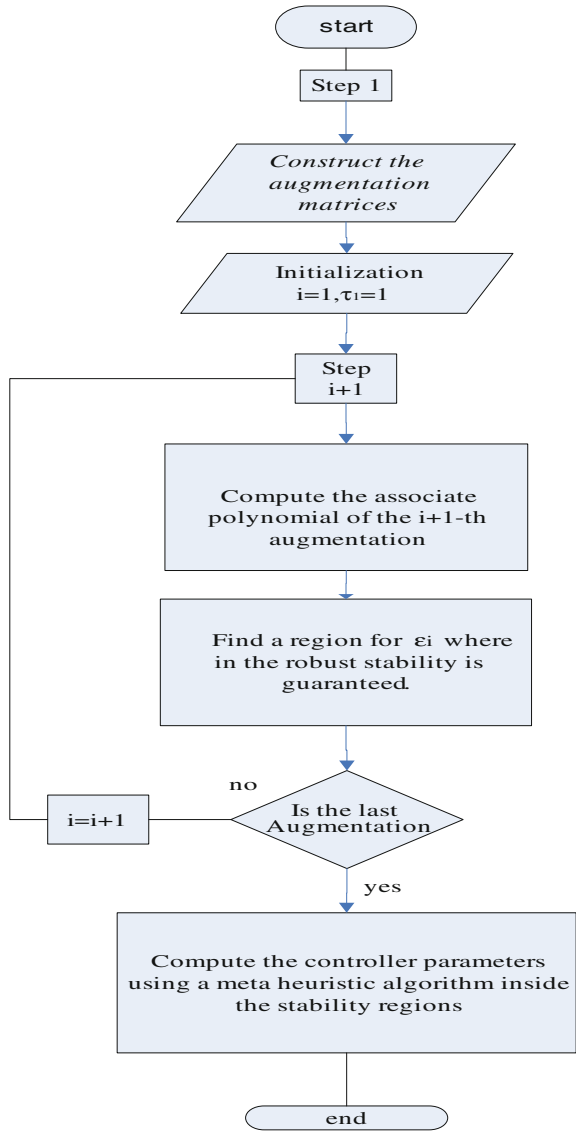
Step 4: Repeat Step 3 until $i \leq n+r+1-\xi$.

Step 5: Using the metaheuristic finite step algorithm presented in [13] in the stability regions $M_i (i = 1, \dots, n+r+1-\xi)$ and for all values of the uncertain parameters, compute the best values for $\varepsilon_1, \dots, \varepsilon_{n+r-1-\xi}$ under performance criteria.

Step 6 (Derivation of the gain vector): Compute the robust tracking controller from the relation:

$$f = \left[\frac{T}{\varepsilon_{n+r+1-\xi}} \left([\varepsilon_{n+r+1-\xi} \dots \varepsilon_1 \ 1] \begin{bmatrix} 0_{1 \times (n+r+1-\xi)} \\ I_{n+r+1-\xi} \end{bmatrix} \right)^T \right]^T.$$

Fig. 5.1 Algorithm flow chart



5.5 Robust Control for Position Tracking of a Hydraulic Actuator

5.5.1 Actuator Model

Consider a double acting servo valve and piston actuator. The linearized differential equations that describe the actuator—valve dynamics can be formulated as follows [9]:

$$\dot{x}_p(t) = v_p(t), \quad \dot{v}_p(t) = \frac{1}{m} [AP_L(t) - bv_p(t) - F_L(t)], \quad (5.16a)$$

$$\dot{P}_L(t) = \frac{4\beta}{V} [K_f x_v(t) - K_{tp} P_L(t) - Av_p(t)]. \quad (5.16b)$$

where v_p is the piston velocity, x_p is the piston position, P_L is the hydraulic pressure across the actuator piston, F_L is the external load disturbance and x_v is the spool valve displacement. The parameters A , m , β , b and V are: the piston surface area, the mass of the load, the effective bulk modulus of the hydraulic fluid, the viscous damping coefficient and the total volume of hydraulic oil in the piston chamber and the connecting lines, respectively. The coefficients K_f and K_{tp} arise from the linearization of the servo valve load flow and the leakage flow.

The valve displacement is usually produced by a solenoid (electrohydraulic valve) actuated by the input voltage $v_{in}(t)$ of the solenoid. The transfer function of a solenoid can be approximated by the servo valve spool position gain denoted by k_u . Using (5.16) the following linear system with uncertain structure is derived in state space form:

$$\dot{x}(t) = A_0(q)x(t) + B_0(q)v_{in}(t) + D_0F_L(t), \quad y(t) = C_0x(t), \quad (5.17a)$$

$$x(t) = [x_p(t) \ v_p(t) \ P_L(t)]^T, \quad C_0 = [1 \ 0 \ 0], \quad (5.17b)$$

$$A_0(q) = \begin{bmatrix} 0 & 1 & 0 \\ 0 & -b/m & A/m \\ 0 & -4q_1A/V & -4q_1q_2/V \end{bmatrix}, \quad B_0(q) = \begin{bmatrix} 0 \\ 0 \\ 4q_1q_3k_u/V \end{bmatrix}, \quad D_0 = \begin{bmatrix} 0 \\ -1/m \\ 0 \end{bmatrix} \quad (5.17c)$$

The parameter $q_1 = \beta$ is an uncertain parameter since the effective bulk modulus of the hydraulic fluid changes due to temperature, pressure and entrained air or water fluctuations. The parameter $q_2 = K_{tp}$ changes due to migration of the system's operating point and the parameter $q_3 = K_f$ changes due to migration of the system operating point and to loss in the effective area of the actuator piston seal, due to wear [9]. The vector $q = [q_1 \ q_2 \ q_3] \in \mathbb{Q}$ is the uncertain vector and \mathbb{Q} is the domain of uncertainty. The nominal values of the system parameters are shown in Table 5.2 and the expected range of variations of the uncertain system parameters is shown in Table 5.1 [9].

Table 5.1 Expected range of variations of the uncertain parameters

Symbol	Minimum values	Nominal values	Maximum values
$\beta(\text{Pa})$	550×10^6	689×10^6	895×10^6
$K_{tp} (\text{m}^3/\text{Pas})$	0	0	9.5×10^{-11}
$K_f (\text{m}^2/\text{sec})$	1.02	1.02	1.76

Table 5.2 Nominal values for the hydraulic actuator's parameters

	Definition	Nominal values
V	Volume of hydraulic oil in the piston chamber	$468/100^3 \text{ m}^3$
A	Piston surface area	$633/1000^2 \text{ m}^2$
β	Effective bulk modulus	$689 \times 10^6 \text{ Pa}$
K_{tp}	Total flow pressure coefficient	$0 \text{ m}^3/\text{Pa-s}$
b	Viscous damping coefficient	$1,000 \text{ Nm}^{-1}\text{s}$
m	Load mass	12 Kg
k_u	Servo valve spool position gain	$0.0406 \times 10^{-3} \text{ m/V}$
K_f	Servo valve gain	$1.02 \text{ m}^2/\text{sec}$

5.5.2 Robust Tracking Controller

In this section a robust tracking arbitrary controller for asymptotic tracking of the piston position will be designed. According to (5.2) the reference output model is derived for $r = 2$ to be:

$$\dot{x}_r(t) = A_r x_r(t), y_r(t) = c_r x_r(t), x_{r,0} = \begin{bmatrix} x_{r,01} \\ x_{r,02} \end{bmatrix}$$

where $A_r = \begin{bmatrix} 0 & 1 \\ -d_2 & -d_1 \end{bmatrix}$ and $c_r = [1 \ 0]$.

According to definitions of Sect. 5.2 the following augmented system is introduced

$$\frac{d}{dt} \tilde{x}(t) = \tilde{A}(q) \tilde{x} + \tilde{b}(q) \tilde{u}(t)$$

where $\tilde{x}(t) = [\varepsilon(t) \ \varepsilon^{(1)}(t) \ z(t)]^T$ and where

$$\tilde{A}(q) = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ -d_2 & -d_1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -b/m & A/m \\ 0 & 0 & 0 & -4q_1 A/V & -4q_1 q_2/V \end{bmatrix}, \quad \tilde{b}(q) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 4q_1 q_3 k_v/V \end{bmatrix}$$

Apply the static state feedback law: $\tilde{u} = f\tilde{x}$ with $f = [f_{11} \ f_{12} \ f_{21} \ f_{22} \ f_{23}]$. The aforementioned controller can be produced by the original input signal $u(t)$ using the following state space form:

$$\dot{x}_c(t) = A_c x_c(t) + b_c e(t), \quad v(t) = c_c x_c(t) \quad (5.18)$$

where $A_c = \begin{bmatrix} -d_1 & 0 \\ -d_2 & 0 \end{bmatrix}$, $b_c = \begin{bmatrix} f_{1,2} \\ f_{1,1} \end{bmatrix}$, $c_c = [1 \ 0]$ and $u(t) = v(t) + f_2 x(t)$ with $f_2 = [f_{21} \ f_{22} \ f_{23}]$.

The augmented system closed loop characteristic uncertain polynomial is:

$$p_{cl}(s, q_1, q_2, f) = s^5 + \gamma_0 s^4 + \gamma_1 s^3 + \gamma_2 s^2 + \gamma_3 s^1 + \gamma_4 \quad (5.19)$$

where

$$\gamma_0(q, f) = d_1 + [4mq_1(q_2 - f_{23}k_u q_3) + bV]/mV,$$

$$\gamma_1(q, f) = \frac{1}{mV}(4A^2q_1 - 4Af_{22}k_u q_1 q_3 + 4bq_1(q_2 - f_{23}k_u q_3) + 4d_1mq_1(q_2 - f_{23}k_u q_3) + bd_1V + d_2mV)$$

$$\gamma_2(q, f) = \frac{1}{mV}(4A^2d_1q_1 - 4A(f_{21} + d_1f_{22})k_u q_1 q_3 + 4d_2mq_1(q_2 - f_{23}k_u q_3) + b(4d_1q_1(q_2 - f_{23}k_u q_3) + d_2V))$$

$$\gamma_3(q, f) = \frac{1}{mV}(4q_1(bd_2(q_2 - f_{23}k_u q_3) + A^2d_2 - A(f_{12} + d_1f_{21} + d_2f_{22})k_u q_3)),$$

$$\gamma_4(q, f) = -[4A(f_{11} + d_2f_{21})k_u q_1 q_3]/mV.$$

According to (5.13) and (5.14) define

$$\tilde{a}(q) = [1 \ \tilde{a}_1 \ \tilde{a}_2 \ \tilde{a}_3 \ \tilde{a}_4 \ \tilde{a}_5] \quad (5.20)$$

$$\tilde{a}_1 = d_1 + \frac{b}{m} + \frac{4q_1q_2}{V}, \quad \tilde{a}_5 = 0$$

$$\tilde{a}_2 = \frac{4A^2q_1 + 4bq_1q_2 + 4d_1mq_1q_2 + bd_1V + d_2mV}{mV}$$

$$\tilde{a}_3 = \frac{4A^2d_1q_1 + 4bd_1q_1q_2 + 4d_2mq_1q_2 + bd_2V}{mV}, \quad \tilde{a}_4 = \frac{4d_2q_1(A^2 + bq_2)}{mV}$$

$$\left[-\tilde{\Omega}(q)\right]^T = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \tilde{\omega}_{25} \\ 0 & 0 & 0 & \tilde{\omega}_{34} & \tilde{\omega}_{35} \\ 0 & 0 & \tilde{\omega}_{43} & \tilde{\omega}_{44} & \tilde{\omega}_{45} \\ 0 & \tilde{\omega}_{52} & \tilde{\omega}_{53} & \tilde{\omega}_{54} & \tilde{\omega}_{55} \\ \tilde{\omega}_{61} & 0 & \tilde{\omega}_{63} & 0 & 0 \end{bmatrix} \quad (5.21)$$

where $\tilde{\omega}_{61} = \tilde{\omega}_{52} = \tilde{\omega}_{43} = \tilde{\omega}_{34} = -\frac{4Ak_u q_1 q_3}{mV}$, $\tilde{\omega}_{25} = -\frac{4k_u q_1 q_3}{V}$, $\tilde{\omega}_{53} = \tilde{\omega}_{44} = -\frac{4Ad_1 k_u q_1 q_3}{mV}$, $\tilde{\omega}_{35} = -\frac{4k_u(b+d_1m)q_1 q_3}{mV}$, $\tilde{\omega}_{55} = -\frac{4bd_2 k_u q_1 q_3}{mV}$, $\tilde{\omega}_{63} = \tilde{\omega}_{54} = -\frac{4Ad_2 k_u q_1 q_3}{mV}$, $\tilde{\omega}_{45} = -\frac{4k_u(bd_1+d_2m)q_1 q_3}{mV}$.

According aforementioned definitions the augmented closed loop characteristic polynomial (5.18) can equivalently be expressed as follows:

$$\tilde{p}_{cl}(s, q, f) = [s^5 \ s^4 \ s^3 \ s^2 \ s^1 \ s^0] A^{**}(q) [1 \ f_{11} \ f_{12} \ f_{21} \ f_{22} \ f_{23}]^T \quad (5.22)$$

where

$$A^{**}(q) = [\tilde{a}^T \ -\tilde{\Omega}^T] \quad (5.23)$$

Let $T = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 \end{bmatrix}$ and choose the following 6×6 row submatrix of $A^{**}(q)T$:

$$A^*(q) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ \phi_{21} & \phi_{22} & 0 & 0 & 0 & 0 \\ \phi_{31} & \phi_{32} & \phi_{33} & 0 & 0 & 0 \\ \phi_{41} & \phi_{42} & \phi_{43} & \phi_{44} & 0 & 0 \\ \phi_{51} & \phi_{52} & \phi_{53} & \phi_{54} & \phi_{55} & 0 \\ 0 & 0 & 0 & \phi_{64} & 0 & \phi_{66} \end{bmatrix} \quad (5.24)$$

where $\phi_{21} = \tilde{a}_0$, $\phi_{31}(q) = \tilde{a}_1$, $\phi_{41} = \tilde{a}_2$, $\phi_{51} = \tilde{a}_3$, $\phi_{22} = -\tilde{\omega}_{25}$, $\phi_{32} = -\tilde{\omega}_{35}$, $\phi_{42} = -\tilde{\omega}_{45}$, $\phi_{52} = -\tilde{\omega}_{55}$, $\phi_{33} = -\tilde{\omega}_{34}$, $\phi_{43} = -\tilde{\omega}_{44}$, $\phi_{53} = -\tilde{\omega}_{54}$, $\phi_{44} = -\tilde{\omega}_{43}$, $\phi_{54} = -\tilde{\omega}_{53}$, $\phi_{64} = -\tilde{\omega}_{63}$, $\phi_{55} = -\tilde{\omega}_{52}$, $\phi_{66} = -\tilde{\omega}_{61}$.

Theorem 3 The problem of robust output command tracking for the uncertain hydraulic model (5.17) via the controller (5.18) is always solvable.

Proof Condition (i) can easily be verified. The matrix $A^*(q)$ is positive antisymmetric. It can be constructed using the five positive up augmentations (ϕ_{66} , ϕ_{55} , ϕ_{44} , ϕ_{33} and ϕ_{22} are positive numbers for all the values of the uncertainties):

$$\Phi_1(q) \rightarrow \Phi_2(q) \rightarrow \Phi_3(q) \rightarrow \Phi_4(q) \rightarrow \Phi_5(q) \rightarrow \Phi_6(q) = A^*(q)$$

where $\Phi_1(q) = \phi_{66}$, $\Phi_2(q) = \begin{bmatrix} \phi_{56} & 0 \\ 0 & \phi_{66} \end{bmatrix}$, $\Phi_3(q) = \begin{bmatrix} \phi_{44} & 0 & 0 \\ \phi_{54} & \phi_{55} & 0 \\ \phi_{64} & 0 & \phi_{66} \end{bmatrix}$, $\Phi_4(q) =$

$$\begin{bmatrix} \phi_{33} & 0 & 0 & 0 \\ \phi_{43} & \phi_{44} & 0 & 0 \\ \phi_{53} & \phi_{54} & \phi_{55} & 0 \\ 0 & \phi_{64} & 0 & \phi_{66} \end{bmatrix}, \quad \Phi_5(q) = \begin{bmatrix} \phi_{22} & 0 & 0 & 0 & 0 \\ \phi_{32} & \phi_{33} & 0 & 0 & 0 \\ \phi_{42} & \phi_{43} & \phi_{44} & 0 & 0 \\ \phi_{52} & \phi_{53} & \phi_{54} & \phi_{55} & 0 \\ 0 & 0 & \phi_{64} & 0 & \phi_{66} \end{bmatrix}.$$

The vector $\bar{c}(q) = \Phi_1(q)$ is a Hurwitz invariant core since the associate polynomial of $\bar{c}(q)$ ($[\bar{c}(q)]^T$) is positive Hurwitz invariant. Hence, condition (ii) of Theorem 1 is satisfied. For reference and disturbance signals of sinusoidal form, condition of Theorem 2 is also satisfied.

5.5.3 An Analytic Algorithm for the Computation of the Controller Parameters

Using a reference input of the form $y_r(t) = 0.02 \sin[0.2t]$ ($d_1 = 0, d_2 = 0.04, x_{r,01} = 0, x_{r,02} = 0.004$) and the algorithm presented in Sect. 5.4 the following data are derived:

Stability Regions: $\varepsilon_1 \in M_1 = [0.25, 0.55]$, $\varepsilon_2 \in M_2 = [0.2, 0.3]$, $\varepsilon_3 \in M_3 = [0.01, 0.015]$, $\varepsilon_4 \in M_4 = [6 \times 10^{-9}, 7 \times 10^{-9}]$, $\varepsilon_5 \in [0.0006, 0.0008]$

Values of the Parameters $\varepsilon_1, \varepsilon_2, \varepsilon_3, \varepsilon_4, \varepsilon_5$: $\varepsilon_1 = 0.5$, $\varepsilon_2 = 0.25$, $\varepsilon_3 = 0.012$, $\varepsilon_4 = 6 \times 10^{-9}$ and $\varepsilon_5 = 0.0007$.

Derivation of the Controller Parameters: $f_{11} = -1428.57$, $f_{12} = -714.286$, $f_{22} = -17.1429$, $f_{21} = -357.143$, $f_{23} = -8.57143 * 10^{-6}$.

5.5.4 Simulation Results

Using Tables 5.2 and 5.1 and for a reference signal and external disturbance as in Figs. 5.2 and 5.3, the closed loop performance is illustrated in Figs. 5.4, 5.5, and 5.6 and the control signal is illustrated in Fig. 5.7. (Dotted line, Solid line, Dashed line is for minimum, intermediary, and maximum value of the uncertainties).

Fig. 5.2 Reference signal

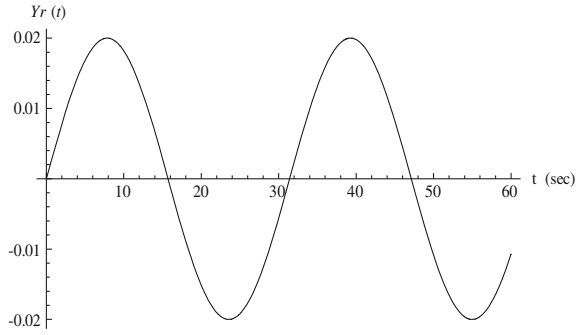


Fig. 5.3 External disturbance

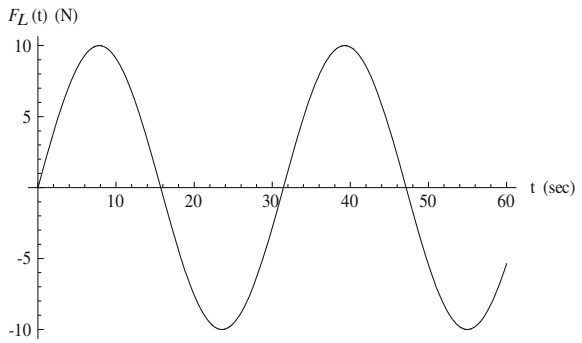


Fig. 5.4 The piston position

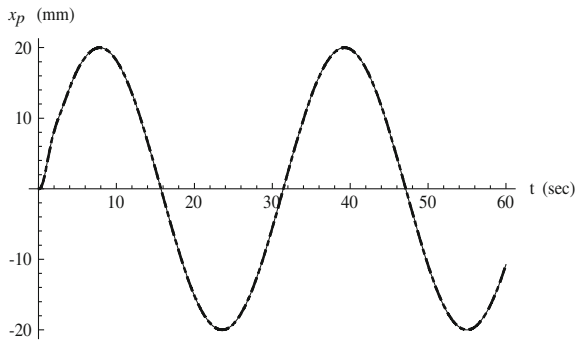


Fig. 5.5 The piston velocity

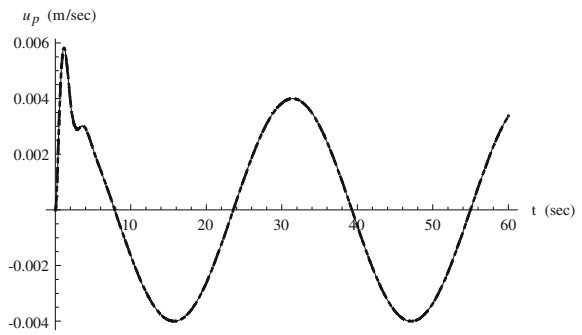
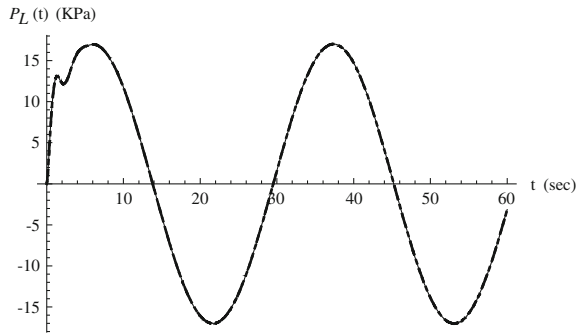
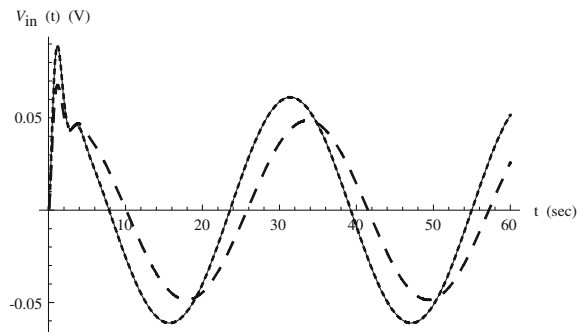


Fig. 5.6 Hydraulic pressure**Fig. 5.7** Input voltage of the solenoid

5.6 Conclusions

A finite step Heuristic Algorithm is proposed in order to solve the problem of robust Internal Model control for uncertain systems. Sufficient conditions have been derived and a finite step algorithm has been proposed for fast and easy computation of the controller parameters. The results are successfully applied to a hydraulic actuator.

Acknowledgments The present research is implemented through the Operational Program “Education and Lifelong Learning” and is co-financed by the European Union (European Social Fund) and Greek national funds (Archimedes III-Strengthening Research Groups in Technological Education, NSRF 2007–2013).

References

1. Chen, C.T.: Linear System Theory and Design. Holt, Rinehart and Winston, New York (1984)
2. Horowitz, I.M.: Synthesis Feedback Systems. Academic Press, New York (1963)
3. Dorf, R.C., Bishop, R.H.: Modern Control Systems. 9th edn. Prentice Hall, Upper Saddle River (2001)
4. Goodwin, G.C., Graebe, S.F., Salgado M.E.: Control System Design. Prentice Hall, Upper Saddle River (2001)

5. Corless, M.J., Leitmann, G., Ryan, E.P.: Tracking in the Presence of Bounded Uncertainties, Presented at the 4th International Conference on Control Theory, Cambridge, UK (1984)
6. Skarpetis, M.G., Koumboulis, F.N., Ntellis, A.: Robust asymptotic output tracking for four-wheel-steering vehicles. In: IEEE International Conference on Mechatronics, (ICM 2006), 3–5 July 2006, Budapest, Hungary, pp. 488–492 (2006)
7. Skarpetis, M.G., Koumboulis, F.N., Ntellis, A.: Robust tracking and disturbance attenuation controllers for automatic steering. In: 14th IEEE Mediterranean Conference on Control and Automation (MED'06), 28–30 June 2006, Ancona, Italy (2006)
8. Skarpetis, M.G., Koumboulis, F.N., Tzamtzi, M.P.: Robust control techniques for hydraulic actuators. In: Proceedings of 15th MED Conference on Control and Automation, Athens, Greece (2007)
9. Karpenko, M., Shapchri, N.: Fault—tolerant control of a servohydraulic positioning system with crossport leakage. *IEEE Trans. Contr. Syst. Technol.* **13**, 155–161 (2005)
10. Koumboulis, F.N., Skarpetis, M.G., Tzamtzi, M.P.: Robust PI controllers for command following with application to an electropneumatic actuator. In: Proceedings of the 14th Mediterranean Conference on Control Automation, Ancona, Italy (2006)
11. Koumboulis, F.N., Skarpetis, M.G., Tzamtzi, M.P.: Robust PID controller design with application to a flight actuator. In: Proceedings of the 32nd Annual Conference of the IEEE Industrial Electronics Society (IECON'06), pp. 4725–4730 (2006)
12. Koumboulis, F.N., Skarpetis, M.G., Mertzios, B.G.: Robust regional stabilization of an electropneumatic actuator. *IEEE Proc. Part D Control Theory Appl.* **145**, 226–230 (1998)
13. Koumboulis, F.N., Tzamtzi, M. P.: A metaheuristic approach for controller design of multivariable processes. In: 12th IEEE International Conference on Emerging Technologies and Factory Automation, 25–28 Sept. 2007, Patras, Greece, pp. 1429–1432 (2007)
14. Skarpetis, M.G., Koumboulis, F.N., Ntellis, A.: Robust arbitrary reference command tracking with application to hydraulic actuators. In: Proceedings of the 9th International Conference on Informatics in Control Automation and Robotics, pp. 94–102 (2012)
15. Wei, K., Barmish, R.: Making a polynomial Hurwitz invariant by choice of feedback gain. *Int. J. Contr.* **50**, 1025–1038 (1989)
16. Koumboulis, F.N., Skarpetis, M.G.: Input–Output decoupling for linear systems with non-linear uncertain structure. *J. Franklin Inst.* **333(B)**, 593–624 (1996)
17. Koumboulis, F.N., Skarpetis, M.G.: Robust triangular decoupling with application to 4WS cars. *IEEE Trans. Autom. Control* **45**, 344–352 (2000)

Chapter 6

A Multi-Signal Variant for the GPU-Based Parallelization of Growing Self-Organizing Networks

Giacomo Parigi, Angelo Stramieri, Danilo Pau and Marco Piastra

Abstract Among the many possible approaches for the parallelization of self-organizing networks, and in particular of *growing* self-organizing networks, perhaps the most common one is producing an optimized, parallel implementation of the standard sequential algorithms reported in the literature. In this chapter we explore an alternative approach, based on a new algorithm variant specifically designed to match the features of the large-scale, fine-grained parallelism of GPUs, in which multiple input signals are processed at once. Comparative tests have been performed, using both parallel and sequential implementations of the new algorithm variant, in particular for a growing self-organizing network that reconstructs surfaces from point clouds. The experimental results show that this approach allows harnessing in a more effective way the intrinsic parallelism that the self-organizing networks algorithms seem intuitively to suggest, obtaining better performances even with networks of smaller size.

Keywords Growing self-organizing networks · Graphics processing unit · Parallelism · Surface reconstruction · Topology preservation

6.1 Introduction

From a general point of view a self-organizing network is composed by units that adapt themselves, through limited and local interactions, to input signals from some predefined space. In most cases a topology is defined among these units by a set of

G. Parigi (✉) · A. Stramieri · M. Piastra
Computer Vision and Multimedia Lab, University of Pavia, Via Ferrata 1, 27100 Pavia, PV, Italy
e-mail: giacomo.parigi@gmail.com

D. Pau
Advanced System Technology, STMicroelectronics, Via Olivetti 2,
20864 Agrate Brianza, MB, Italy

binary connections. At first sight, the adaptation process may look inherently parallel, since each unit follows the same predetermined behavior and in many cases, as long as two units are sufficiently far away in the network, they do not interact in any way.

Nonetheless, most of the algorithms in the literature are described as sequential procedures, in the sense that input signals are submitted one by one to the network and processed each in a sequential way. This means that, in most cases, also units will be adapted sequentially, one after the other, even when they can be considered as mutually independent, i.e. with input signals that are sufficiently distant in the input space.

In a typical algorithm, each input signal has to be compared to all units in the network, in order to find the closest one and adapt the latter and its neighbors to the input signal. For reasons that will be described in detail later on, this operation is dominant in terms of execution time, and is therefore the obvious focus for parallel implementation. In this respect, two main methods emerge: *data partitioning* methods, in which the input signals are partitioned across parallel tasks, whereby each task involves the entire network and processes just one input signal; *network partitioning* methods, in which the units of the network are partitioned across parallel tasks, whereby each task considers all input signals but only in relation to the units belonging to its partition. These two approaches are thoroughly examined in [2] for the parallelization of Kohonen's self-organizing map [3]. In particular, in the former work, a data partitioning approach is described for the batch version of the algorithm, and a network partitioning approach for the on-line version of the algorithm, in both cases for an SP2 parallel computer.

As a matter of fact, perhaps the most common approach for the parallel implementation of self-organizing networks described in the literature (see for instance [4–7]), is to adapt the network-partitioning method to the standard, sequential version of the algorithm.

In an effort to better harness the “latent parallelism” of self-organizing networks, a new algorithm variant for *growing* self-organizing networks is proposed in this chapter. In this *multi-signal* algorithm variant, a number of signals are submitted to the network and elaborated at once during each iteration. This variant is explicitly intended for a data-partitioning approach to parallelization, which, as described in [2], may offer “the potential for much greater scalability, since the parallel granularity is determined by the volume of data, which is potentially very large”. In particular the new algorithm focuses on *growing* self-organizing networks and this entails dealing with some further functional aspects, that are not present in the algorithm for self-organizing maps considered in [2]. This aspect will be described in Sect. 6.2.

The new multi-signal algorithm has been designed to match the features of the large-scale, fine-grained parallelism of GPUs (Graphics Processing Units). Beside its computational capabilities, this hardware has gained a large popularity due to the lower costs compared to those of more traditional high-performance computing solutions. For instance, in [8], GPUs have been defined “probably today's most powerful computational hardware for the dollar”.

The GPU-based implementation of the multi-signal variant, has shown good performances in all the tests performed, reaching noticeable speed-ups even for smaller

networks. In addition, the new multi-signal algorithm has shown some interesting differences w.r.t. the standard single-signal algorithm: in the tests performed, the multi-signal algorithm always required less input signals to reach termination than the single-signal counterpart. These aspects will be further discussed in Sect. 6.3.

6.2 Methods

6.2.1 Growing Self-Organizing Networks

In the discussion that follows, we consider as reference a network in which each unit is associated to a *reference vector* in the input space, and a topology is defined by a set of binary connections between the units. These connections also define the local topology, or *neighborhood*, of each unit. In a self-organizing network units are progressively adapted during the learning process. In addition, *growing* self-organizing networks, like GNG [9], GWR [10] and SOAM [1] have the following characteristics:

- during the learning process the number of units varies, and can both grow and shrink;
- the topology of connections between units varies as well, since connections are both created and destroyed during the learning process.

In general, the learning process of a growing self-organizing network can be described as a basic iteration, which is repeated until some convergence criterion is met:

1. *Sample*

Generate at random one input signal ξ with probability $P(\xi)$. Usually the support of $P(\xi)$ coincide with the *region of interest*, i.e. the region of input space to be considered.

2. *Find Winners*

Compute the distances $\|\xi - \mathbf{w}_i\|$ between each reference vector and the input signal and find the k -nearest units. In most cases $k = 2$, i.e. the nearest (*winner*) and second-nearest units are searched for.

3. *Update the Network*

Create a new connection between the winner and the second-nearest unit, if not existing, or reset the existing one.¹

Adapt the reference vector of the winner unit and of its topological neighbors, with a law of the type:

$$\begin{aligned}\Delta \mathbf{w}_b &= \varepsilon_b \|\xi - \mathbf{w}_b\|, \\ \Delta \mathbf{w}_i &= \varepsilon_i \eta(i, b) \|\xi - \mathbf{w}_i\|,\end{aligned}\tag{6.1}$$

¹ An *aging* mechanism is also applied to connections (see for instance [9]).

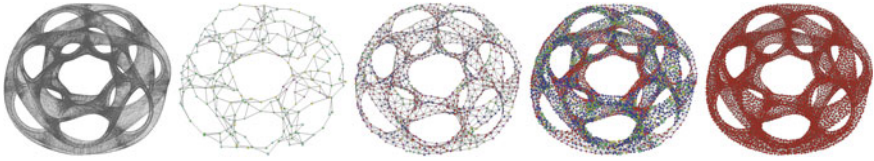


Fig. 6.1 The SOAM [1] reconstructs a surface from the point cloud on *left*. At the end, all units converge to the same stable state

where \mathbf{w}_b is the reference vector of the winner and \mathbf{w}_i are the reference vectors of the neighboring units. $\varepsilon_b, \varepsilon_i \in [0, 1]$ are the *learning rates*, with $\varepsilon_b \gg \varepsilon_i$. The function $\eta(i, b) \leq 1$ determines how neighboring units are adapted. In most cases $\eta(i, b) = 1$ if units b and i are connected and 0 otherwise. During this phase, new units can be created and old units can be removed, with methods that may vary depending on the specific algorithm.

The three steps above are iterated until some convergence criterion is met: typically, in most self-organizing networks, including growing ones, this criterion is a threshold on the overall quantization error, i.e. the mean squared distance between input signals and the corresponding winners. For the purposes of this work we adopted the SOAM algorithm, that has a termination criterion which does not depend on a threshold. In the SOAM algorithm, in fact, the learning process terminates when all units have reached a local topology consistent with that of a surface and therefore the network represents the triangulation of the surface that has to be reconstructed from input signals (see Fig. 6.1). The clear termination criterion in the SOAM algorithm is fundamental for comparing the performances and the different behaviors of the two variants of the algorithm, i.e. single-signal and multi-signal, and their implementations.

The methods adopted for adding and removing units during the *Update* phase greatly vary depending on the algorithm. In GNG, for example, new units are inserted at regular intervals, in the neighborhood of the unit i that has accumulated the largest mean squared error. In contrast, in GWR new units are added whenever the distance between the winner unit and the input signal ξ is greater than a predefined *insertion threshold*. The new unit is positioned in proximity of the winner and the network topology is updated. The SOAM algorithm is similar to the GWR, with the difference that the insertion threshold may vary during the learning process, in order to reflect the *local feature size* (LFS) of the surface being reconstructed (see Sect. 6.3.1).

In terms of time complexity, the *Find Winners* phase is dominant in general. In fact, assuming that the number k of nearest neighbors is constant and small, the *Find Winners* phase has $\mathcal{O}(N)$ time complexity, where N is the total number of units. Although the complexity of the *Update* phase may greatly vary depending on how the function is defined (see for instance the Neural Gas algorithm [11]), as a matter of fact in most growing self-organizing networks, including the SOAM algorithm, this update is local and limited to the connected neighbors of the winner, so that the *Update* phase can be assumed to have $\mathcal{O}(1)$ time complexity. Furthermore, in this

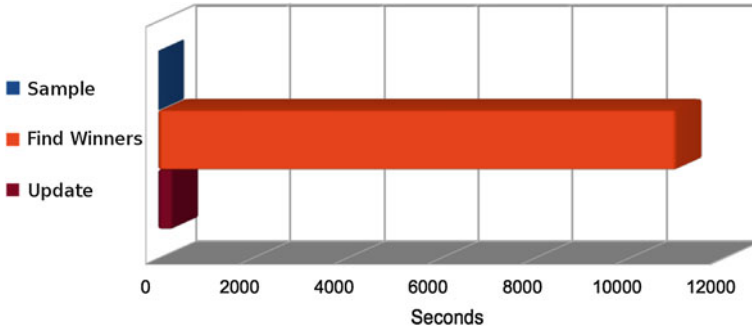


Fig. 6.2 Single-phase time to convergence of the SOAM algorithm (average values on the whole test set)

discussion, we will not consider the *Sample* phase in detail: sampling methods, in fact, are application-dependent and not necessarily under the control of the algorithm.

The dominance of the *Find Winners* phase in terms of time complexity is confirmed by experiments. The graph in Fig. 6.2 shows the mean values obtained from the experiments described in Sect. 6.3. These results are in line with the ones reported in the literature (see for example [4] for a detailed analysis), in that the percentage of the execution time spent in the *Find Winners* phase remains as low as 50–60% of the total execution time as long as the network remains small (i.e. 250–500 units), but grows rapidly to 95% and more as the network size increases and more signals are processed.

6.2.2 The Multi-Signal Variant

In the *multi-signal* variant proposed here, at each iteration $m \gg 1$ signals are considered at once. The basic iteration hence becomes:

1. *Sample*

Generate at random m input signals ξ_1, \dots, ξ_m according to the probability distribution $P(\xi)$, as described before.

2. *Find Winners*

For each signal ξ_j , compute the distances $\|\xi_j - \mathbf{w}_i\|$ between each reference vector and the input signal and find the k -nearest units.

3. *Update the Network*

For each signal ξ_j , perform the update operations specified in the previous section.

The first two phases in the above iteration pose no particular problems, since all the involved operations performed for each signal are mutually independent. In contrast, during the *Update* phase, the operations performed for different signals may collide. In particular three kinds of collision can occur:

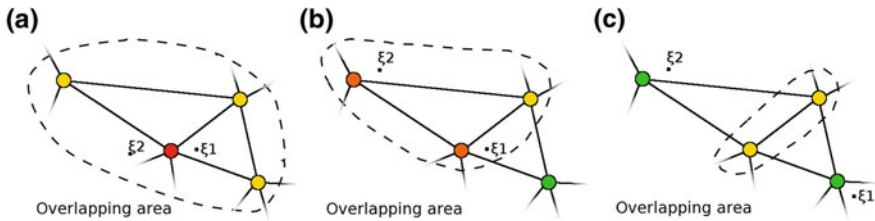


Fig. 6.3 Collision caused by two different input signals ξ_1 and ξ_2 . In (a) the two signals share the same winner, hence all direct neighbors. In (b) and (c) the winner is different, but the neighbors are shared. All three cases would lead to colliding adaptations

Adapt Position. Two or more signals lead to the adaptation of the same unit in the network. Collisions of this kind are usually not isolated, since the collision can happen both for the winner and for the neighboring units, as described in Fig. 6.3.

Modify Neighborhood. Two or more signals lead to modifying the neighborhood of the same unit. This may be caused by either the insertion/removal of units or the creation/removal of edges.

Insert Edge. Two or more signals lead to the insertion of the same edge.

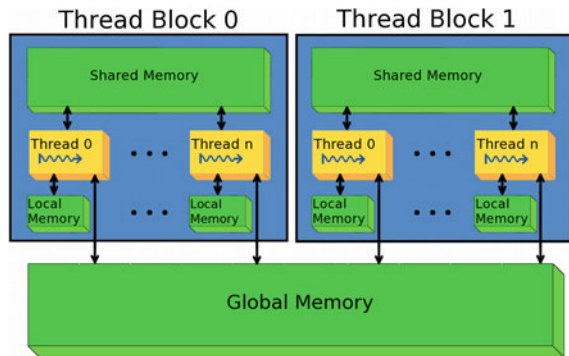
In the multi-signal variant presented here, the main concern in choosing the method for managing the above collisions is maintaining a coherent behavior with respect to the single-signal algorithm, and allow an unbiased comparison of the performances. At the same time, the method must be simple enough. The solution adopted is using an implicit *lock* on the winner unit: no two or more input signals having the same winner can cause any of the colliding modifications to be performed, as only the first incoming signal, in a random order, will produce the corresponding effect, while other signals for the same winner will just be *discarded*.

Collisions apart, the behavior of the new variant is *different* from the original, single-signal algorithm. In the experiments described in Sect. 6.3, in fact, the multi-signal variant always required a smaller number of signals to reach convergence, regardless of the implementation. This aspect will be discussed in more detail in Sect. 6.3.2.

6.2.3 Graphics Processing Units

Graphics Processing Units (GPUs) are specialized and optimized for graphic applications, and are typically mounted on dedicated boards with private onboard memories. During these last years, GPUs have evolved into general-purpose parallel execution machines [12]. Until not many years ago, in fact, the only available programming interfaces (API) for GPUs were very specific, forcing the programmer to translate the task into the graphic primitives provided. Gradually, many general-purpose API for parallel computing have emerged, which are suitable for GPUs as well. This set

Fig. 6.4 Standard GPU memory hierarchy



of API includes, for instance, RapidMind [13], PeakStream [14] or the programming systems owned by NVIDIA and AMD, respectively CUDA (Compute Unified Device Architecture) [15] and CTM (Close to Metal) [16], together with proposed vendor-independent standards like OpenCL [17].

Albeit with some differences, all these API adopt the general model of *stream computing*: data elements are organized in *streams*, which are ordered sets of data; a set of streams is processed by the same *kernel*, intended as a set of functions to be computed in parallel, and produces another set of streams as output. Each kernel is executed on a set of GPU cores in the form of concurrent *threads*, each executing the same program on a different stream of data. Within a kernel, threads are grouped into blocks and each block is executed in sync. In case of branching of the execution, the block is partitioned in two: all the threads on the first branch are executed in parallel and then the same is done for all the threads in the second branch. This general model of parallel execution is often called SIMT (single-instruction multiple-thread) or SPMD (single-program multiple-data); compared to the older SIMD, it allows greater flexibility in the flow of different threads, although at the cost of a certain degree of serialization, depending on the program. This means that, although independent thread executions are possible, blocks of coherent threads with limited branching will make better use of the GPU's hardware.

In typical GPU architectures, onboard and on-chip memories are organized in a hierarchy (Fig. 6.4): *global* memory, i.e. accessible by all threads in execution, *shared* memory, i.e. a faster cache memory dedicated to each single thread block and *local* memory and/or registers, which are private to each thread.

Another noteworthy feature of modern GPUs is the wide-bandwidth access to onboard memory, on the order of 10x the memory access bandwidth on typical PC platforms. To achieve best performances, however, memory accesses by different threads should be made aligned and coherent, in order to *coalesce* them into fewer, parallel accesses addressing larger blocks of memory. Incoherent accesses, on the other hand, must be divided into a larger number of sequential memory operations. One of the aspects that make GPU programming still quite complex is that, in most cases, the hierarchy of levels of memory, in particular the shared memory, has to be

managed explicitly by the programmer. In return, this explicit management is often an opportunity for further optimizations and better performances.

6.2.4 GPU-Based Parallel Implementation of the Single-Signal Algorithm

In the work presented here we did not produce a parallel implementation of the single-signal algorithm, but we relied on the results reported in the literature, instead.

For the parallelization of (single-signal) growing self-organizing network algorithms, a very common approach is applying the well-known *map-reduce* pattern, which can be easily parallelized into a two-step method, to the dominant *Find Winners* phase. In the first step of the *map-reduce* approach, i.e. the *map* operation, the distance from each unit to the input signal is computed in parallel. In the second step, i.e. the *reduce* operation, the set of previously computed distances is iteratively reduced by comparing subsets in parallel, until the k shortest distances are found. In passing, Buck et al. describe GPU reductions in more detail in the context of the Brook programming language [18], while Harris does it in [19] for the CUDA language. The map-reduce pattern has been studied in general [20] and applied to the search of k nearest neighbors (k -NN) [21]. More specifically this approach has been used for the parallelization of the GNG algorithm (see [5] and [4]) and of the Parameter-Less SOM (see [7]).

The *map-reduce* approach, however, implies a one-to-one correspondence between network units and GPU threads in the *map* phase, which becomes even lower in the *reduce* phase. This fact becomes a substantial limitation for the parallelization of growing self-organizing networks, which usually start with a very small number of units and grow progressively during the execution. As reported (see [5]), unless the network contains at least 500–1000 units, the sequential execution on a CPU can be faster than the parallel one. To cope with this problem, a hybrid technique has been proposed (see [5] and [4]): switching the execution from CPU to GPU only when the network is sufficiently large and the latter hardware is expected to perform better. Nevertheless, even with this hybrid solution, the maximum level of parallelization that can be attained is bound to the number of units in the network.

6.2.5 GPU-Based Parallel Implementation of the Multi-Signal Variant

For the GPU-based parallel implementation of the algorithm, the main advantage of the multi-signal variant is that the level of parallelism is bound only by the number of signals submitted to the network at each iteration. Furthermore this same level of parallelism can be maintained across entire kernels, since no reduction takes place.

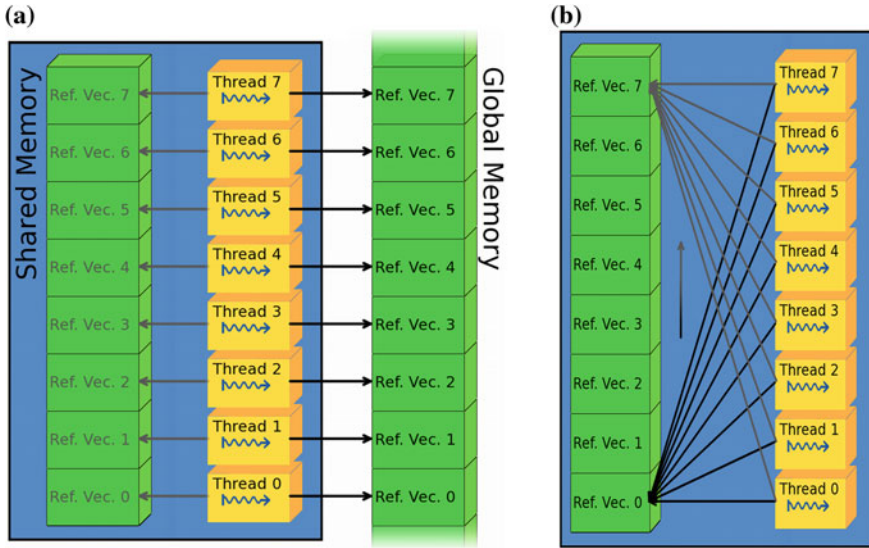


Fig. 6.5 The two steps of the Find Winners phase in the parallel implementation. **a** Parallel load, **b** Sequential scan

The only limitation of this variant comes from the collisions that can occur during the *Update* phase, as explained in Sect. 6.2.2. Nevertheless, if the parallel implementation focuses on the dominant *Find Winners* phase, there is in practice no upper limit for the level of parallelism, beyond that of the hardware.

In the *kernel* that has been realized for the *Find Winners* phase, each thread is assigned to an input signal. The execution is divided in two steps (see Fig. 6.5): first, all threads in a block load a contiguous batch of reference vectors in the shared memory with a *coalesced* access; second, all threads compute the distances from the reference vectors to the signal through a sequential scan of the shared memory, in which all threads read the same reference vector in sync. From the point of view of GPU-based parallelization, this allows harnessing the faster and smaller *shared memory* in order to accelerate the access to the *global memory*, i.e. where the whole set of reference vectors is stored.

6.3 Experimental Validation

6.3.1 Methods of Comparison

All the experiments described in this section have been performed with the SOAM algorithm, in four different implementations (see below), applied to the same tasks

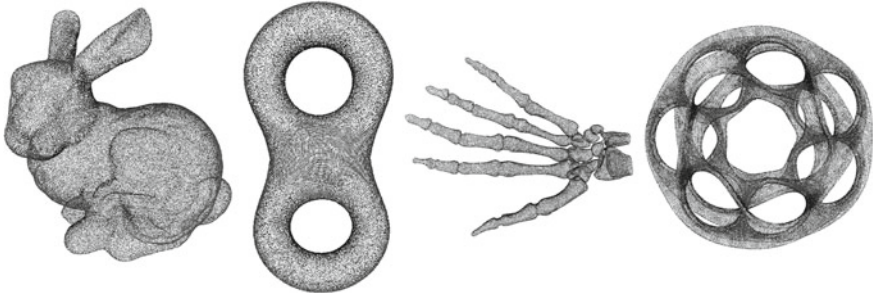


Fig. 6.6 The four point-clouds used in the test phase

of surface reconstruction from point clouds. In each experiment, the point cloud was taken from a triangular mesh and sampled with uniform probability distribution $P(\xi)$.

Four different meshes have been used, each having different *topological* and *geometrical complexity*. More precisely, we consider two measures, one for each type of complexity: the *genus* of the surface [22], i.e. the number of holes enclosed by it, and the *local feature size* (LFS), defined in each point x of the surface as the minimal distance to the medial axis [23]. In this perspective, a ‘simple’ mesh has genus zero or very low and almost constant LFS values, while a ‘complex’ mesh has higher genus and LFS values that vary widely across different areas.

The four meshes used in the experiments are well-known benchmarks for surface reconstruction (Fig. 6.6):

- *Stanford Bunny*. It has genus 0 and some non-negligible variations in the LFS values that make it non-trivial.
- *Eight* (also called *double torus*). It has genus 2 and relatively constant LFS values almost everywhere.
- *Skeleton Hand*. It has genus 5 and widely variable LFS values, that in many areas, e.g. close to the wrist, become considerably low.
- *Heptoroid*. It has genus 22, and low and variable LFS values over the surface.

Obviously, there is no *a priori* guarantee that a parallel algorithm should be faster than a highly-optimized sequential one. Therefore we chose to implement also a single-signal variant of the algorithm in which the crucial *Find Winners* phase is improved through the use of a *hash indexing* method, similar to that used in molecular dynamics [24].

The hash index is constructed by defining a grid of cubes of fixed size inside an axis-parallel bounding box that contains all the input signals in the input space. The hash index of both signals and reference vectors, which live in the same space, is obtained from the 3D coordinates. Whenever an input signal is selected, the search for the reference vectors of the winner and the second nearest units is first performed on the same cube where the input signal resides, together with its 26 adjacent cubes. If this search fails, the exhaustive search is performed instead. Even if this method is

slightly approximate, in that in a few extreme cases it may produce different results from the exhaustive search, it yields a substantial speed-up, as will be discussed in Sect. 6.3.3. In addition, being an hash method, the maintenance of the index, which is performed in the *Update* phase, does not affect performances in a significant way.

Four different implementations of the SOAM algorithm have been used for the experiments:

- *Single-signal*. A reference implementation of the single-signal SOAM algorithm in C.
- *Indexed*. The same single-signal algorithm, but using an hash index for the *Find Winners* phase.
- *Multi-signal*. A reference implementation in C of the multi-signal variant of the algorithm, as described in Sects. 6.2.2 and 6.2.5 but without any actual parallelization, in terms of execution.
- *GPU-Based*. An implementation in C and NVIDIA C/CUDA of the multi-signal variant of the algorithm, with actual hardware parallelization.

The tests have been performed on a Dell Precision T3400 workstation, with a NVidia GeForce GT 440, i.e. an entry-level GPU based on the *Fermi* architecture. The operating system was MS Windows Vista *Business* SP2 and all the programs have been compiled with MS Visual C++ Express 2010, with the CUDA SDK version 4.0.

All the shared input parameters have been set to the same values for all the tests for the four different implementations, while implementation-specific parameters, such as the level of parallelism or the index cube size, have been tuned specifically for maximum performances. Among the shared input parameters, only the crucial *insertion threshold* has been tuned for each mesh, for the reasons described in [1], while every other parameter value has been kept constant for all the four meshes.

In order to avoid discarding an excessive number of signals in the *Update* phase, in all parallel implementations the level of parallelism m at each iteration, i.e. the number of signals processed in the iteration, is set to the minimum power of two greater than the current number of units in the network. The maximum level of parallelism has been set to 8192.

Tables 6.1, 6.2, 6.3 and 6.4, at the end of this section, show the numerical results obtained from the experiments. As it can be seen, for each input mesh, each different implementation reaches a final configuration which can be either different or very different, e.g. for the skeleton hand, in terms of number of units and connections. Note that multi-signal and GPU-based implementations reach exactly the same final configuration, since they are meant to replicate the same behavior by design.

As expected, there are substantial differences also for execution times. In the tables these are measured as *total time to convergence* and *time per signal*, and the detail figures are reported for each of the three phases. Time per signal is a measure of the raw performances that can be obtained with each implementation, while time to convergence is the combined result of the implementation *and* the different behavior that each implementation induces, as it will be explained in the next sections.

Table 6.1 Execution time and statistics on the Stanford Bunny data-set

Algorithm version	Single-signal	Indexed	Multi-signal	GPU-based
<i>Network configuration at convergence</i>				
Iterations	620,000	616,000	1,296	1,296
Signals	620,000	616,000	580,656	580,656
Discarded signals	0	0	319,054	319,054
Units	330	332	347	347
Connections	984	990	1,035	1,035
<i>Time to convergence</i>				
Total time	4.9530	3.369	3.893	2.059
Sample	0.460	0.048	0.009	0.016
Find winners	2.610	1.233	2.448	0.699
Update	1.883	2.088	1.436	1.344
<i>Time per signal</i>				
Time per signal	7.9887×10^{-06}	5.4692×10^{-06}	6.7045×10^{-06}	3.5460×10^{-06}
Find winners	4.2097×10^{-06}	2.0016×10^{-06}	4.2159×10^{-06}	1.2038×10^{-06}

Table 6.2 Execution time and statistics on the Eight data-set

Algorithm version	Single-signal	Indexed	Multi-signal	GPU-based
<i>Network configuration at convergence</i>				
Iterations	1,100,000	1,100,000	1,128	1,128
Signals	1,100,000	1,100,000	1,100,110	1,100,110
Discarded signals	0	0	562,277	562,277
Units	656	649	658	658
Connections	1,974	1,953	1,980	1,980
<i>Time to convergence</i>				
Total time	12.3540	5.5690	11.6070	3.8690
Sample	0.0150	0.0480	0.0620	0.1410
Find winners	8.8600	2.8220	8.5060	0.7650
Update	3.4790	2.6990	3.0390	2.9630
<i>Time per signal</i>				
Time per signal	1.1231×10^{-05}	5.0627×10^{-06}	1.0551×10^{-05}	3.5169×10^{-06}
Find winners	8.0545×10^{-06}	2.5655×10^{-06}	7.7320×10^{-06}	6.9539×10^{-07}

6.3.2 Behavior of the Multi-Signal Algorithm

The first five lines of Tables 6.1, 6.2, 6.3 and 6.4, highlight an aspect that is worth some further discussion, in particular for the *Single-signal* and the *Multi-signal* implementations.

Regardless of the hardware parallelization, the *Multi-signal* variant always needs a substantially lower number of input signals than the *Single-signal* one to converge. This difference becomes even more evident if the discarded signals are not counted for, approaching a ratio of one to four as the mesh becomes more complex.

Table 6.3 Execution time and statistics on the Hand data-set

Algorithm version	Single-signal	Indexed	Multi-signal	GPU-based
<i>Network configuration at convergence</i>				
Iterations	202,988,000	213,800,000	10.264	10.264
Signals	202,988,000	213,800,000	81.092.912	81.092.912
Discarded signals	0	0	33.432.622	33.432.622
Units	5,669	5,766	8.884	8.884
Connections	17,037	17,328	26.688	26.688
<i>Time to convergence</i>				
Total time	18, 548.4937	5, 337.2451	12, 422.3738	872.0250
Sample	9.4050	35.9820	8.6120	8.0480
Find winners	17, 763.1367	4, 127.8511	11, 789.8398	241.1750
Update	775.9520	1, 173.4120	623.9220	622.8020
<i>Time per signal</i>				
Time per signal	9.1377×10^{-05}	2.4964×10^{-05}	1.5319×10^{-04}	1.0753×10^{-05}
Find winners	8.7508×10^{-05}	1.9307×10^{-05}	1.4539×10^{-04}	2.9741×10^{-06}

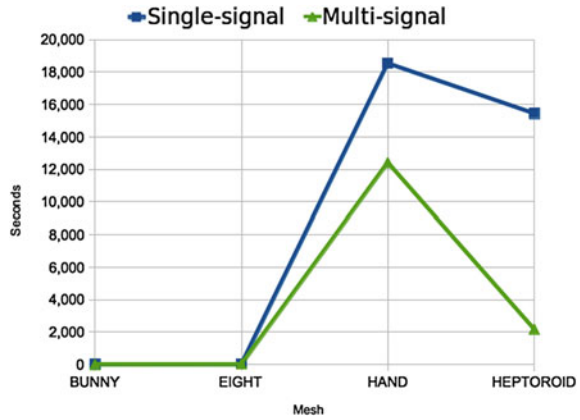
Table 6.4 Execution time and statistics on the Heptoroid data-set

Algorithm version	Single-signal	Indexed	Multi-signal	GPU-based
<i>Network configuration at convergence</i>				
Iterations	20,714,000	23,684,000	1,244	1,244
Signals	20,714,000	23,684,000	7,683,554	7,683,554
Discarded signals	0	0	2,262,969	2,262,969
Units	14,183	13,937	15,638	15,638
Connections	42,675	41,937	47,040	47,040
<i>Time to convergence</i>				
Total time	15, 449.2950	950.0250	2, 172.8009	119.6530
Sample	6.9570	3.4550	0.8010	0.5630
Find winners	15, 294.3330	780.5370	2, 089.6169	34.2640
Update	148.0050	166.0330	82.3830	84.8260
<i>Time per signal</i>				
Time per signal	7.4584×10^{-04}	4.0113×10^{-05}	2.8279×10^{-04}	1.5573×10^{-05}
Find winners	7.3836×10^{-04}	3.2956×10^{-05}	2.7196×10^{-04}	4.4594×10^{-06}

The decrease in the number of signals to convergence is attained despite the growth in the number of units and connections reached in the final configuration.

Figure 6.7 compares the times to convergence of the *Single-signal* and *Multi-signal* implementations. The performances of *Multi-signal* implementation are always better than its *Single-signal* counterpart, a difference that becomes more substantial as the complexity of the mesh increases. Overall, this means that the extra load due to the increase in the number of both units and connections is outbalanced by the decrease in the number of signals needed to reach convergence.

Fig. 6.7 Time to convergence of the Single-signal and Multi-signal implementations



In a possible explanation, the multi-signal variant has a better inherently distributed behavior than the original variant. In fact, in each iteration of the multi-signal variant, a randomly scattered set of units is updated ‘simultaneously’, whereas in the single-signal variant only the winner unit and its direct neighbors are updated. Apparently, the more distributed update leads to a more effective use of the input signals, yielding faster convergence. This aspect, however, requires further investigation.

6.3.3 GPU-Based Implementation Performances

Figure 6.8 shows a summary of the total times to convergence for the *Single-signal*, *Indexed* and *GPU-based* implementations respectively, for the two most complex meshes, with detail figure per each phase. Remarkably, in the *GPU-based* implementation, the *Find Winners* phase ceases to be dominant, and the *Update* phase becomes the most time-consuming. This means that in this implementation any further optimization of the *Find Winners* phase is less relevant unless the execution of the *Update* phase is sped up in turn.

More in detail, Fig. 6.9a shows the average times per signal spent in the *Find Winners* phase for each of the three implementations. Clearly, these times grow as the number of the units in the network becomes larger. Figure 6.9b compares the speed-up factors in average time per signal for the *Indexed* and *GPU-based* implementations with respect to the *Single-signal* one. As expected, these factors also grow with the number of units in the network, since the hash index in the *Indexed* implementation becomes more effective, while an higher level of parallelism can be achieved in the *GPU-based* implementation. Overall, the speed-up factor for the GPU-based implementation reaches 165x on the *Heptoroid* mesh.

Figure 6.10a shows the total times to convergence. These results show how the performances of the SOAM algorithm depend on the variation in the LFS values

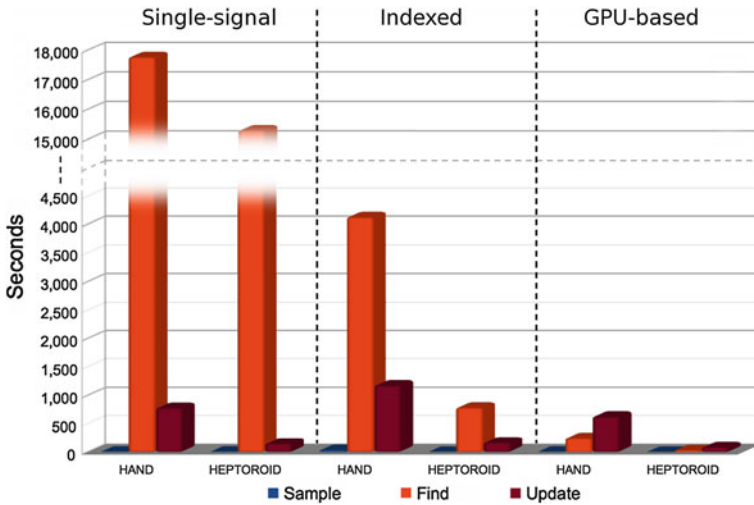


Fig. 6.8 Single-phase time to convergence for the two more complex meshes in the test set

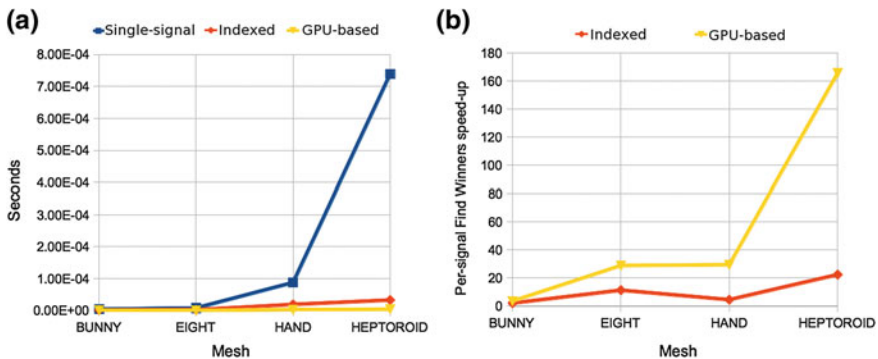


Fig. 6.9 Per-signal performances. **a** Times per signal in the Find Winner phase for the three implementations. **b** Speed-up factors for the Find Winners phase time per signal compared to the Single-signal implementation

(see Sect. 6.3.1): in fact, the skeleton hand always requires the longest time to convergence, regardless of the implementation. Figure 6.10b shows the speed-up factors for the time to convergence, for the *Indexed* and *GPU-based* implementations, again with respect to the *Single-signal* one. These factors grow with the number of units in the network, and are the combined results of the implementation and of the behavior induced.

For all input meshes, the total times to convergence for the *GPU-based* implementation are much lower than the ones for the *Single-signal* implementation. Speed-ups vary from 2.5x (bunny) to 129x (heptoroid), as the complexity of the mesh increases and the size of the reconstructed network grows. In particular, the results obtained

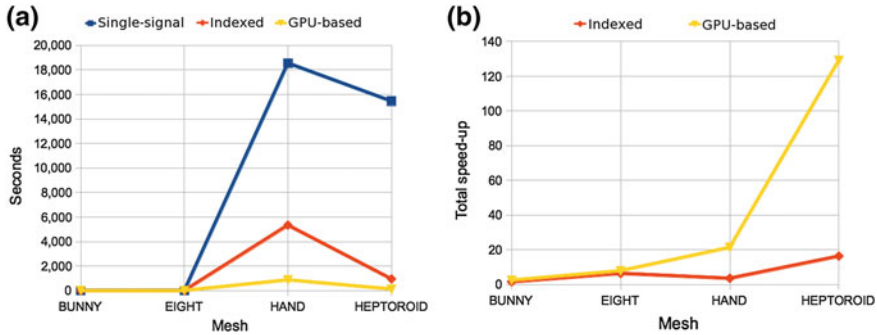


Fig. 6.10 Global performances. **a** Times to convergence for the three implementations. **b** Speed-up factors for the times to convergence compared to the Single-signal implementation

with the *Stanford Bunny*, given in Table 6.1, show non negligible speed-up factors in both the total time to convergence (2.5x) and the time per signal (3.5x), despite that the network contains only 330–347 units at most.

This result is particularly relevant if compared to other GPU-based parallel implementations of growing self-organizing networks (see for example [5]), for which it is reported that the GPU execution produces noticeable speed-ups only when the networks contain no less than 500–1000 units.

The *Indexed* implementation of the algorithm also obtains noticeable speed-ups on all test cases. Nonetheless, as shown in Fig. 6.8, the *Find Winners* phase is still dominant, although with *Stanford bunny* and *Eight* the *Update* times become comparable.

6.4 Conclusions and Future Developments

In this chapter we examined the parallelization of growing self-organizing networks by proposing a multi-signal variant of the original algorithm adopted, in order to increase its parallel scalability.

In particular, the experiments show that this multi-signal variant adapts naturally to the GPU architecture in that, besides the advantages deriving from the careful management of hierarchical memory through perfectly coalesced memory accesses, it leads to a better usage of the high number of cores by allowing very high numbers of concurrent threads.

A further interesting, and somehow unexpected, result of the experiments is that, hardware parallelization apart, the overall behavior of the multi-signal variant is significantly different from the original, single-signal one. The multi-signal variant of the algorithm, in fact, seems to better deal with complex meshes, by requiring a smaller number of signals in order to reach network convergence. This aspect needs to be further investigated, possibly with more specific and extensive experiments.

The parallelization described in this chapter is limited to the dominant *Find Winners* phase and, according to the experimental results, can successfully make it less time-consuming than the *Update* phase. This means that future developments of the strategy proposed should aim to the parallelization of the *Update* phase as well, in order to further improve on performances. This requires some care however, as the *collisions* among threads trying to update the data structures involved, must be managed with care.

References

1. Piasra, M.: Self-organizing adaptive map: autonomous learning of curves and surfaces from point samples. *Neural Netw.* **41**, 96–112 (2012)
2. Lawrence, R., Almasi, G., Rushmeier, H.: A scalable parallel algorithm for self-organizing maps with applications to sparse data mining problems. *Data Min. Knowl. Disc.* **3**, 171–195 (1999)
3. Kohonen, T.: *Self-Organizing Maps*. Springer series in information sciences, vol. 30. Springer, Berlin (2001)
4. Orts, S., Garcia-Rodriguez, J., Viejo, D., Cazorla, M., Morell, V.: Gpgpu implementation of growing neural gas: application to 3d scene reconstruction. *J. Parallel Distrib. Comput.* **72**(10), 1361–1372 (2012)
5. García-Rodríguez, J., Angelopoulou, A., Morell, V., Orts, S., Psarrou, A., García-Chamizo, J.: Fast image representation with gpu-based growing neural gas. *Adv. Comput. Intell., Lect. Notes Comput. Sci* **6692**, 58–65 (2011)
6. Luo, Z., Liu, H., Wu, X.: Artificial neural network computation on graphic process unit. In: *Proceedings of the IEEE International Joint Conference on Neural Networks, IJCNN'05. IEEE 2005*, vol. 1, pp. 622–626 (2005)
7. Campbell, A., Berglund, E., Streit, A.: Graphics hardware implementation of the parameter-less self-organising map. In: *Intelligent Data Engineering and Automated Learning-IDEAL 2005*, pp. 5–14 (2005)
8. Owens, J., Luebke, D., Govindaraju, N., Harris, M., Krüger, J., Lefohn, A., Purcell, T.: A survey of general-purpose computation on graphics hardware. *Comput. Graphics Forum* **26**, 80–113 (2007) (Wiley Online Library)
9. Fritzke, B.: *A growing neural gas network learns topologies*. In: *Advances in Neural Information Processing Systems 7*, MIT Press (1995)
10. Marsland, S., Shapiro, J., Nehmzow, U.: A self-organising network that grows when required. *Neural Netw.* **15**, 1041–1058 (2002)
11. Martinetz, T., Schulten, K.: Topology representing networks. *Neural Netw.* **7**, 507–522 (1994)
12. Owens, J., Houston, M., Luebke, D., Green, S., Stone, J., Phillips, J.: Gpu computing. *Proc. IEEE* **96**, 879–899 (2008)
13. McCool, M.: Data-parallel programming on the cell be and the gpu using the rapidmind development platform. In: *GSPx Multicore Applications Conference*. vol. 9. (2006)
14. Papakipos, M.: *The Peakstream Platform: High-Productivity Software Development for Multi-Core Processors*. PeakStream Inc., Redwood City (2007)
15. NVIDIA Corporation: *CUDA C programming guide, version 4.0*. Santa Clara (2011)
16. Hensley, J.: AMD CTM overview. In: *ACM SIGGRAPH 2007 courses*, p. 7. ACM, New York (2007)
17. Stone, J., Gohara, D., Shi, G.: Opencl: a parallel programming standard for heterogeneous computing systems. *Comput. Sci. Eng.* **12**, 66 (2010)
18. Buck, I., Foley, T., Horn, D., Sugerman, J., Fatahalian, K., Houston, M., Hanrahan, P.: Brook for gpus: stream computing on graphics hardware. *ACM Trans. Graph.* **23**, 777–786 (2004)

19. Harris, M.: Optimizing parallel reduction in cuda. CUDA SDK Whitepaper (2007)
20. Liu, S., Flach, P., Cristianini, N.: Generic multiplicative methods for implementing machine learning algorithms on mapreduce. Arxiv, preprint arXiv:1111.2111 (2011)
21. Zhang, C., Li, F., Jestes, J.: Efficient parallel knn joins for large data in mapreduce. In: Proceedings of 15th International Conference on Extending Database Technology (EDBT 2012). (2012)
22. Edelsbrunner, H.: GGeometry and topology for mesh generation. Cambridge Monographs on Applied and Computational Mathematics, vol. 7. Cambridge University Press, Cambridge (2001)
23. Amenta, N., Bern, M.: Surface reconstruction by voronoi filtering. *Discrete Comput. Geom.* **22**, 481–504 (1999)
24. Hockney, R.W., Eastwood, J.W.: *Computer Simulation Using Particles*. Taylor & Francis Inc., Bristol (1988)

Part II
Robotics and Automation

Chapter 7

Office Delivery Robot Controlled by Modular Behavior Selection Networks with Planning Capability

Young-Seol Lee and Sung-Bae Cho

Abstract Recently, assistance service using mobile robots becomes one of important issues. Accordingly, studies on controlling the mobile robots are spreading all over the world. In this line of research, we propose a hybrid architecture based on hierarchical planning of modular behavior selection networks for generating autonomous behaviors of the office delivery robot. Behavior selection network is suitable for goal-oriented problems, but it is too difficult to design a monolithic behavior network to deal with complex robot control. We decompose it into several smaller behavior modules and construct sequences of the modules considering the sub-goals, the priority in each task and the user feedback. The feasibility of the proposed method is verified on both the Webot simulator and Khepera II robot in an office environment with delivery tasks. Experimental results confirm that a robot can achieve goals and generate module sequences successfully even in unpredictable and changeable situations, and the proposed planning method reduces the elapsed time during tasks by 17.5 %.

Keywords Office delivery robot · Hybrid robot control · Behavior networks

7.1 Introduction

Recent robot studies focus service robots [1] because robot market for service robots will grow exponentially in the near future. Since service robots are investigated and made in order to provide various services, they should be controlled to satisfy

S.-B. Cho (✉) · Y.-S. Lee
Department of Computer Science, Yonsei University, 50 Yonsei-ro,
Seodaemun-gu, Seoul 120-749, Korea
e-mail: sbcho@cs.yonsei.ac.kr

Y.-S. Lee
e-mail: tiras@sclab.yonsei.ac.kr

diverse goals [2]. For this reason, much research for controlling service robot appears. Especially, the mobile robots in the office environment are very helpful for users to conduct routine tasks. Several control structures for the office delivery robots have been proposed with various approaches [3–6].

Despite much effort to control service robot, most of conventional approaches focused on generating behaviors of mobile robots in well-known environments. The conventional planning-based methods can generate the behavior sequences optimized in predefined environments, but have the difficulty of low flexibility in complex environments. On the while, reactive systems can generate primitive behaviors quickly based on environmental stimuli in complex domains [7]. But it is also difficult to create goal-oriented behaviors to perform a user's task robustly. In order to overcome these problems, hybrid behavior generation architectures including the characteristics of the deliberative and reactive systems are proposed. Chung and Williams divided the original problem into several sub-problems to perform plans by reducing the complexity of the problem [4] and Ramachandran and Gupta proposed POMDP-based reinforcement learning for delivery robot [6]. Some reactive methods look like similar to the proposed method that can deal with environmental changes without environmental information. But they have the limitation to achieve only local goals and react to current exceptions without any consideration of global goals.

To work out this problem, some hybrid architectures have been proposed. Milford and Wyeth used different obstacles and experience maps for local and global navigations, respectively [5]. The method used low-level controls for reactive actions that were managed by high-level controls. In this chapter, we present hybrid architecture so that service robots in office environment can provide delivery services to users and use modular design approach based on several behavior networks and planning method, which are regarded as the reactive and deliberative levels, respectively. For the service robot, the behavior-based method is more appropriate because it is more important to achieve goals and maintain autonomy. In this reason, the proposed architecture exploits the behavior networks for autonomous behaviors of the office delivery robot, which have been known as useful in goal-oriented problems [8–11].

This chapter is an updated and extended version of [12] by including additional experiments for changing environment. The rest of this chapter is organized as follows: Sect. 7.2 describes the proposed hybrid architecture based on modular behavior networks and module-sequence planning. Section 7.3 evaluates the proposed architecture in terms of qualitative and quantitative analysis, and Sect. 7.4 concludes this chapter and discusses future works.

7.2 Hybrid Architecture

The proposed architecture for the autonomous office delivery robot to generate behaviors consists of two levels. Lower level is behavior network-based modules to handle temporary environmental changes, and upper level is a deliberative system to control the goals and plans flexibly according to situations. Figure 7.1 shows the proposed

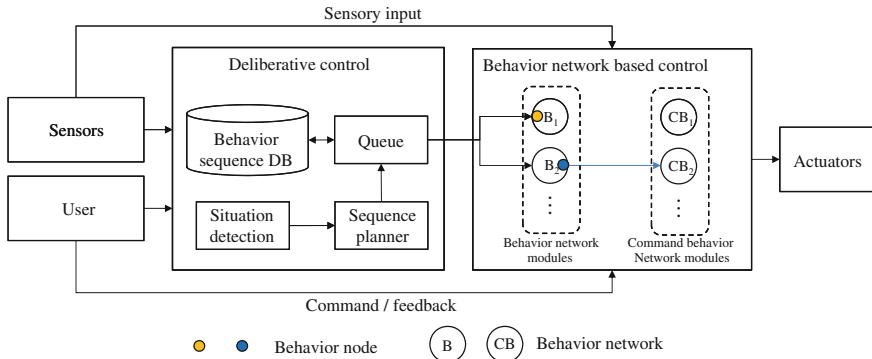


Fig. 7.1 Architecture of the proposed hybrid behavior network system

architecture of the hybrid behavior network system. The behavior network-based control includes the specific behavior networks and the common behavior networks, and the deliberative plan control.

7.2.1 Behavior Network Modules

Unlike the conventional reactive systems, the behavior network can generate behaviors instantly as well as goals. The behavior network enables us solve some simple planning problems. However, as the problem gets more complex, it is difficult to select behaviors accurately with only one monolithic network [13, 14]. In order to overcome this shortcoming, the behavior network is divided into several modules.

The objectives of the modularized behavior networks are as follows.

- The modular behavior network is easier to be designed and reused than one monolithic network [8].
- Confusions which can be occurred when selecting behaviors in one large flat network can be reduced by giving only one goal to each smaller network module [14].

Each module in the proposed architecture has a behavior network oriented to single corresponding goal. The behavior network is used as the method for selecting the most natural and suitable behaviors for the situations. The behavior networks are the model that consists of relationships between behaviors, a goal, and external environment, and selects the most suitable behavior for the current situation.

In the behavior network, behaviors, external environments and internal goals are connected with each other through links. Each behavior contains preconditions, an add list, a delete list and an activation. The preconditions are a set of conditions that must be true in order to execute behaviors. The add list is a set of conditions that are highly likely to be true when behaviors are executed. The delete list is a set of

conditions that are likely to be false when the behavioral entities are executed. The activation represents to what extent the behavioral entity is activated.

The activation energies of behaviors firstly induced from external environments and the goal. The activation of the i th behavior A_i can be presented as follows:

$$A_i = A_i + \sum_n w_e E_{i,n} + \sum_m w_g G_{i,m} (E_{i,n}, G_{i,m} = 0, 1) \quad (7.1)$$

where w_e and w_g are the weights to induce activation energies from environments and goal respectively. $E_{i,n}$ and $G_{i,m}$ represent whether the n th environment element and the m th goal are connected with the i th behavior or not, respectively.

After the first induction, behaviors exchange their activation energies with other behaviors considering the type of their links. The behavior exchange can be presented as follows:

$$A_i = A_i + \sum_n (w_p P_{i,j} + w_s S_{i,j} - w_c C_{i,j}) \quad (i \neq j, P_{i,j}, S_{i,j}, C_{i,j} = 0, 1) \quad (7.2)$$

where w_p , w_s and w_c are the weights to exchange activation energies through predecessor, successor and conflictor links, respectively, and $P_{i,j}$, $S_{i,j}$ and $C_{i,j}$ represent whether the i th and j th behaviors are connected by each type of links, respectively.

A behavior network module consists of one goal, external environments, and behavior nodes. Each module is mapped to a sub-goal from the planning system. If the planning system chooses a single sub-goal to achieve, the corresponding behavior network module is activated and generates behavior sequences. In this chapter, we designed two behavior network modules—go to a room and find objects—and two common modules—navigate and avoid obstacles. Figure 7.2 shows the behavior network modules designed.

7.2.2 Planning of Goal Sequences

In the deliberative control, the system does not plan sequences of all primitive behaviors or trajectories, but plans the sequences of sub-goals to control behavior network modules. Since we designed several small independent behavior modules with sub-goals, they should be controlled explicitly to achieve the global goal. To plan goal sequences, the deliberative module and the behavior network-based modules are connected. Since the behavior networks do not have any information about the map of the environment, it is difficult to perform plans correctly in complex environments. To deal with this, the deliberative module checks accomplishments of sub-goals and controls plans when situations are changed, and the plan in each behavior network module controls only partial behavior sequences to achieve the sub-goal of the corresponding module.

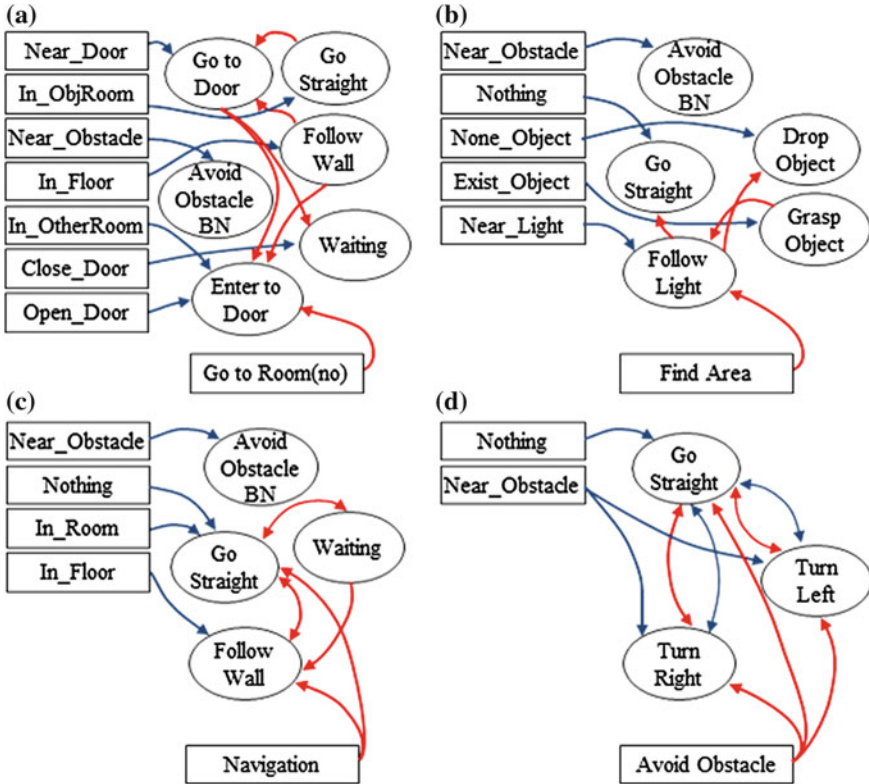


Fig. 7.2 The behavior networks designed

The deliberative control module makes plan by deciding priorities of goal sequences to achieve the global goal and adjusting priorities when exceptions or feedbacks are occurred. The module uses the basic behavior library that includes basic sequences of behaviors required to perform when tasks are given. The library is defined before the usage, and can be modified by the feedbacks of the user. When the user gives tasks, the sequences are planned by using the library and inserted into the queue. At the ‘Check event’ stage, the robot checks changing of situations, and adjusts the sequences.

To plan and adjust the sequences, the priorities of tasks are used. In this chapter, the priority is defined as the deadline of the delivery required by the user. For this process, we define several parameters as follows:

- $C = \{c_i\}$: command set
- $D = \{\{d_i\} : \{d_i\} \in C\}$: decomposed command set
- $Q = \{q_i : q_i = d_l, \dots, d_k, i < \max_{queue}\}$: command queue
- $X = \{Wait, Critical, Minor\}$: user feedback set

Firstly, priorities are determined according to the requested deadline and the order of tasks as shown below:

$$P_{Fix}^i = \frac{(t_{Max} - t_i)}{t_{Max}} \times 10 + (O_{Max} - O_i) \quad (7.3)$$

where t_i and O_i indicate the remaining time and the order of the i th task, respectively. Max means the possible maximum value of the corresponding variable.

Secondly, priorities are adjusted by additionally considering the position of the robot as follows:

$$P_{Dynamic}(q_i) = \begin{cases} P_{Fix}(q_i), & \text{if } From(i) = S \text{ or } t_i < \theta \\ P_{Fix}(q_j), & \text{if } From(i) \neq S \text{ and } \exists j. \ni \cdot From(j) = S \\ f(S), & \text{if } S \in X \end{cases} \quad (7.4)$$

where S is the current state of the robot, $From(i)$ indicates the starting point of the i th task, and $f(S)$ is the priority decided by the feedback.

The sequence queue contains feedbacks from the user. Each of them consists of an index of the user, a type of command, a deadline, a point of departure, and a destination. When the feedback is given, the robot seeks sequences for the corresponding command and puts the sequences into the queue. If there is no relevant sequence in the library, the robot requests feedbacks to the user.

The priorities of behavior modules in the sequence are computed with the order of the task and the deadline by using Eqs. (7.3) and (7.4). Each module is sorted by the priority in the sequence queue. For this job, the queue has information. The front four are input by the user, and next five are used to manage the plan flexibly.

Each task has the segmented sequence with subtasks. For example, a single delivery task is split into the subtask to bring the object from the point of departure and another subtask to move the object to the destination. Each task has a check point that indicates which subtask is performed lastly. The check point enables to adjust the plan flexibly according to the change of situations. The subtask has the sequence of several behavior modules.

Task adjustments are preceded according to the position of the robot as follows:

$$Seq(q_i) = \begin{cases} q_k, & \text{if } \exists Pos_k. \ni \cdot Pos_k = S \text{ and } t_i > \theta \\ q_j \rightarrow d_l, & \text{if } \exists Pos_j. \ni \cdot Pos_j = S \\ \text{and } (q_j \rightarrow d_l = Take \text{ or } q_j \rightarrow d_l = Give \text{ and } \exists obj) \\ q_i, & \text{otherwise} \end{cases} \quad (7.5)$$

where $Seq(q_i)$ indicates the target command to be placed instead of q_i , Pos_i is a set of positions that q_i contains, and $q_i \rightarrow d_l$ is the l th behavior in q_i . For example, the robot may pass the other room not required for the task during the movement from the starting point to the destination. In this case, it searches the task which the robot should fulfill at its current location. If the deadline of the task in progress is greater

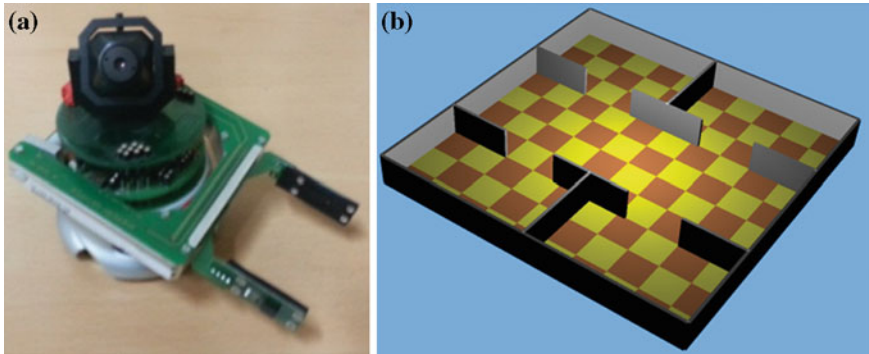


Fig. 7.3 a Khepera II and b Webot simulator

than the threshold, it changes the plan to execute the task found with high priority. Otherwise, it ignores the task found and continues its previous job.

7.3 Experiments

In order to show the usefulness of the proposed architecture, we performed experiments for the office delivery tasks of the mobile robot.

7.3.1 Experimental Setup

The hybrid behavior generation system is applied to the mobile robot, Khepera II, which has a wireless camera sensor, eight infra-red sensors, eight light sensors, one gripper and two motors as shown in Fig. 7.3a. The experiments were performed on both the Webot simulation environment in Fig. 7.3b and a real-world environment. Figure 7.4a shows the real world environment setting and Fig. 7.4b illustrates the constructed real world environment.

For the office delivery tasks, we designed the office environment which includes four rooms and one aisle. The colors of each pair of the door and the room were colored identically; therefore, the robot can recognize each room by referring the color of the corresponding door. If some doors had been closed, we changed colors of them as blacks. Since the robot does not have any information about the environment, it should navigate with only recognized colors of rooms.

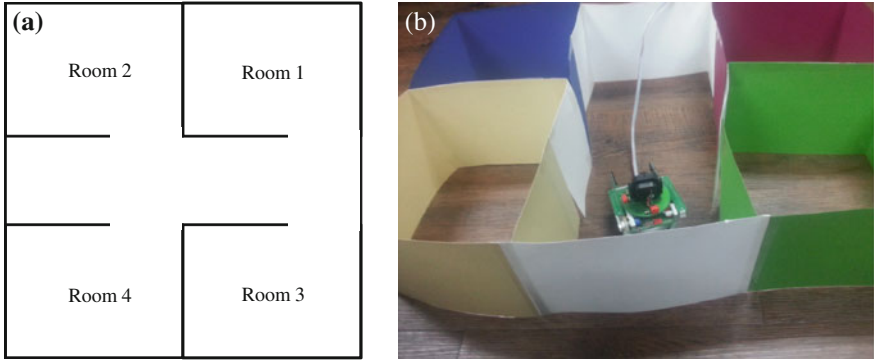


Fig. 7.4 The experimental environment with four rooms and a corridor

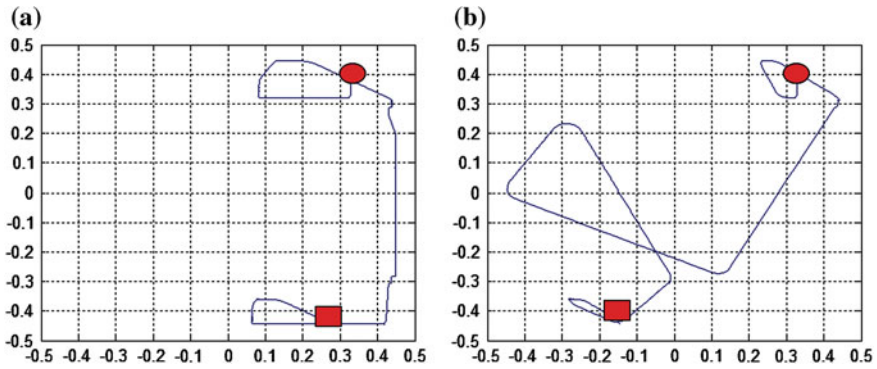


Fig. 7.5 Trajectories of the robot

7.3.2 Analysis of Experimental Result

In this section, we analyzed planned goal sequences from various tasks. We obtained the rates of success and failure after performing all tasks, and analyzed changing of the sequences according to errors and feedbacks from the user.

The task of delivering the object from the specific room A to another room B was given for the experiments. First of all, we obtained the trajectories of the robot during the task. Figure 7.5a and b are the trajectories for the delivery task from the room 3 to the room 1 and the task from the room 4 to the room 1, respectively.

If the robot had been located in the room or at the corridor, it started the behavior module for searching the destination and used camera for sensing since it did not have map information of the environment. When the robot reached the destination room, it followed the light to find the object.

Table 7.1 Given seven delivery tasks

Task	1	2	3	4	5	6	7
Deadline	1	1	2	3	1	2	1
Departure (Room #)	1	3	2	3	1	3	4
Destination (Room #)	2	1	3	1	4	2	1

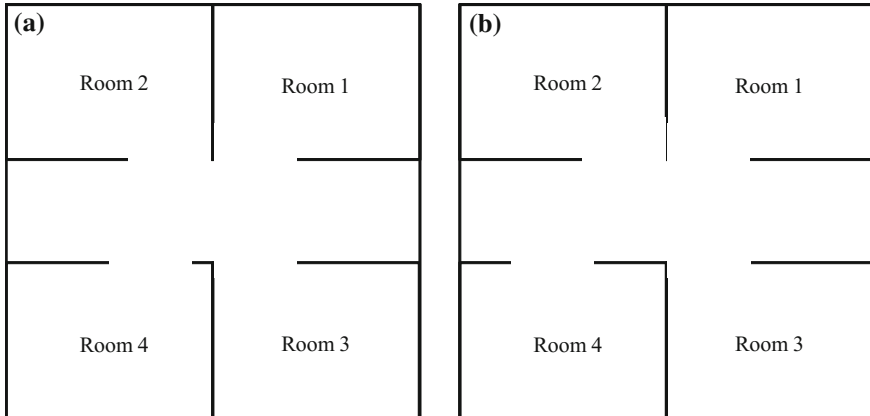


Fig. 7.6 a Original location of door in room 4 b Changed door location of room 4

Additionally, in order to verify the usefulness of the sequence adjusting process, we designed seven delivery tasks shown in Table 7.1. Experiments were conducted both with and without sequence adjustments using the tasks. Sequences of chosen modules and robot’s location were obtained. With sequence adjustment processes, the robot modified its behavior sequence according to its location. If the robot achieved its goal in the certain room, it sought the task which can be started at the room. As the result, it reduced steps wasted at the corridor. The robot finished all the tasks within 3,956 steps without sequence adjustments, but it completed within 3,264 steps, 17.4% reduced, with adjustment processes.

Additionally, we changed the location of the door of room to verify the adaptability of our proposed method to dynamically changing environment as shown in Fig. 7.6. In this example, delivery tasks are performed from room 2 to room 4. Figure 7.7a and b show the examples. In Fig. 7.7a, the door of room 4 is located in $(-0.2, 0.0)$. The location of the door of room 4 in Fig. 7.7b is changed to $(-0.3, 0.0)$. Although the location of door is changed, the delivery task is successfully performed by our proposed method.

For quantitative analysis, we obtained the elapsed time during tasks. We initially located the robot randomly and made it to repeat random delivery tasks 30 times. Table 7.2 shows minimum, average, and maximum steps after tasks. The task from the room 4 to the room 2 took the smallest steps. The maximum steps were taken in

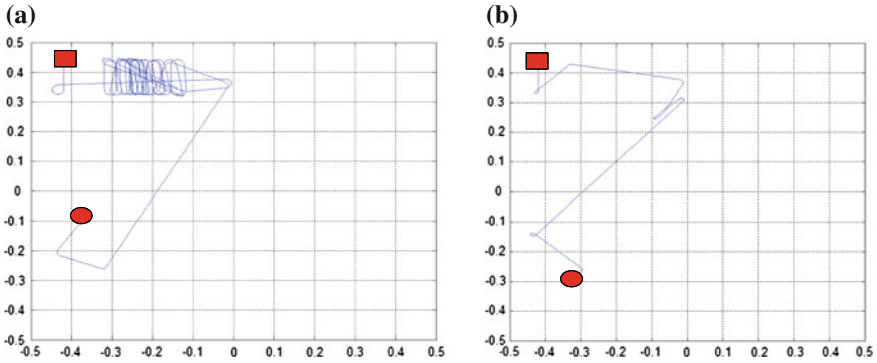


Fig. 7.7 Trajectories from room 2 to room 4 **a** door $(-0.2, 0.0)$ **b** door $(-0.3, 0.0)$

Table 7.2 Minimum, average, and maximum steps after 30 tasks

Minimum	Average	Maximum
804	1,930	5,370

the case that the robot was initially located at the corridor because it took long time to find the target room according to the state of the sensors.

7.4 Concluding Remarks

We presented a hybrid behavior system for an autonomous mobile robot for office delivery tasks. The system is oriented to the behavior network modules which is useful to perform tasks in real-world environments. Moreover, a method for planning is attached to supplement them. The planning system generates and manages overall sequences of behavior modules, and the behavior modules achieve several sub-goals by generating autonomous behaviors quickly.

Experiments were conducted to verify the usefulness of the proposed architecture. We implemented a simple office environment in both the simulator and the real-world with the Khepera II mobile robot, and designed several delivery tasks. As the result, it is confirmed that the robot can achieve the goal even though there are temporary exceptions, and it changes its plan when adjustments are required to complete tasks more efficiently.

For the future works, the method for learning structures of networks and controlling them automatically should be investigated. Moreover, the proposed architecture should be tested on more realistic problems.

Acknowledgments This research was supported by the Original Technology Research Program for Brain Science through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (2010-0018948).

References

1. Park, H.-S., Cho, S.-B.: A modular design of bayesian networks using expert knowledge: context-aware home service robot. *Expert Syst. Appl.* **39**(3), 2629–2642 (2012)
2. Garcia, E., Jimenez, M.A., Santos, P.G., Armada, M.: The evolution of robotics research. *IEEE Robot Autom. Mag.* **14**(1), 90–103 (2007)
3. Beetz, M., Arbuckle, T., Belker, T., Cremers, A.B., Schulz, D., Bennewitz, M., Burgard, W., Hahnel, D., Fox, D., Grosskreutz, H.: Integrated, plan-based control of autonomous robots in human environments. *IEEE Intell. Syst.* **15**(5), 56–65 (2001)
4. Chung, S.H., Williams, B.C.: A Decomposed symbolic approach to reactive planning. Master's Thesis, MIT (2003)
5. Milford, M., Wyeth, G.: Hybrid robot control and SLAM for persistent navigation and mapping. *Rob. Auton. Syst.* **58**(9), 1096–1104 (2010)
6. Ramachandran, D., Gupta, R.: Smoothed sarSa: reinforcement learning for robot delivery tasks. In: *IEEE International Conference on Robotics and Automation*, pp. 2125–2132. IEEE Press, New York (2009)
7. Mataric, M.J.: Using communication to reduce locality in distributed multi-agent learning. *J. Exp. Theor. Artif. Intell.* **10**(3), 357–369 (1998)
8. Nicolescu M.N., Mataric, M.J.: A hierarchical architecture for behavior-based robots. In: *First International Joint Conference on Autonomous Agents and Multi-Agent Systems*, pp. 227–233. ACM, New York (2002)
9. Weigel, T., Gutmann, J.-S., Dietl, M., Kleiner, A., Nebel, B.: CS freiburg: coordinating robots for successful soccer playing. *IEEE Trans. Robot. Autom.* **19**(5), 685–699 (2002)
10. Yoon, J.-W., Cho, S.-B.: A mobile intelligent synthetic character with natural behavior generation. In: *2nd International Conference on Agents and, Artificial Intelligence, ICAART 2010*, pp. 315–318. Valencia, (2010)
11. Lim, S.-S., Yoon, J.-W., Oh, K.-H. Cho, S.-B.: Gesture based dialogue management using behavior network for flexibility of human robot interaction. In: *IEEE International Symposium on Computational Intelligence in Robotics and Automation*, pp. 592–597. IEEE Press, New York (2009)
12. Yoon, J.-W., Cho, S.-B.: Hierarchical planning of modular behaviour networks for office delivery robot. In: *9th International Conference on Informatics in Control, Automation and, Robotics*, pp. 14–20. Rome, (2012).
13. Decuqis, V., Ferber, J.: An extension of maes' action selection mechanism for animats. In: *5th International Conference on Simulation of Adaptive Behavior on From animals to animats 5*, pp. 153–158. MIT Press, Cambridge, (1998)
14. Tyrell, T.: Computational mechanisms for action selection. PhD Thesis, University of Edinburgh (1993)

Chapter 8

Worst-Case Performance Analysis in ℓ_1 -norm for an Automated Heavy Vehicle Platoon

Gábor Rödönyi, Péter Gáspár, József Bokor and László Palkovics

Abstract Based on model set identification and unfalsification, robust performance measured in peak-to-peak gain is analyzed for heterogeneous platoons, inter-vehicle communication delays and actuator uncertainties. The goal is to demonstrate that safe platooning with acceptable performance can be achieved by utilizing the services already available on every commercial heavy truck with automated gearbox. Experimental verification of a three vehicle platoon is also presented.

Keywords Vehicle platoons · Peak-to-peak gain · Performance unfalsification

8.1 Introduction

Safe control of vehicle platoons requires strict guaranteed bounds on inter-vehicle spacing errors. In order to avoid collision the sampled errors are best measured by their ℓ_∞ norm, so the bounds represent the worst-case peaks of the spacing errors. Consistent identification tools are the set membership methods in the ℓ_1 setting, see e.g. [2, 5, 6]. The identified model sets are employed for on-line model

G. Rödönyi (✉) · P. Gáspár · J. Bokor
Systems and Control Laboratory, Computer and Automation Research
Institute of Hungarian Academy of Sciences, Budapest, Hungary
e-mail: rodonyi@sztaki.hu

P. Gáspár
e-mail: gaspar@sztaki.hu

J. Bokor
e-mail: bokor@sztaki.hu

L. Palkovics
Knorr-Bremse Brake-systems GmbH., Budapest, Hungary
e-mail: Laszlo.Palkovics@knorr-bremse.com

validation and a priori analysis of the control performance measured by the worst-case spacing errors.

Controllers for autonomous vehicle platoons usually consist of two levels of feedback controllers. At the lower level a local, vehicle specific controller is responsible for performing acceleration demands. The higher level control law is common for all vehicles, it is designed for satisfying string stability requirements of the entire platoon. Very short safety gaps can be guaranteed under certain constraints on lead vehicle maneuvers, when detailed engine, gearbox and brake system models are available, see, e.g., in references [1, 4, 9]. There is, however, some difficulty in the widespread applicability of these control methods. The required engine/gearbox/brake system models are usually not available and not reliable for all commercial heavy trucks. In addition these controllers try to directly excite the brake cylinder pressures and the throttle valve of the engine, which could also conflict with the existing control units, such as Electronic Brake System (EBS) and Engine Control Unit (ECU).

In this chapter the goal is to explore the performance of an automated vehicle string where, in contrast to the former solutions, only the standardized and general services of the EBS and ECU are used. This work is an extension of the research that was presented in the conference paper [11], where the focus was placed on model set identification problems and the analysis of the spacing error bounds subject to heterogeneity in vehicle dynamics. A method for computing unfalsified performance in order to analyze the effect of actuator uncertainties is also presented. An illustration is shown for the brake system.

In Sect. 8.2 the mathematical model of the platoon is presented. The vehicle model set identification method is provided in Sect. 8.3. The performance of a heterogeneous platoon and the effect of actuator uncertainties are analyzed in Sect. 8.4. The experimental results are shown in Sect. 8.5.

Basic notations. The peak norm of a sequence $u(k)$ is denoted by $\|u\|_\infty = \sup_k |u(k)|$, ℓ_∞ denotes the space of sequences of finite peak norm. The peak-to-peak norm of a system H is defined by $\|H\|_1 = \sup_{u \neq 0} \frac{\|Hu\|_\infty}{\|u\|_\infty}$.

8.2 State-Space Model of Vehicle Platoons

In this section a discrete-time, linear time-varying state-space model for the controlled platoon is briefly summarized.

The longitudinal dynamics of a single vehicle is approximated by the following first order nominal model with sampling time T_s

$$\hat{a}_i(k+1) = \theta_{i1}^* \hat{a}_i(k) + \theta_{i2}^* u_i(k), \quad i = 0, 1, \dots, n \quad (8.1)$$

$$a_i(k) = \hat{a}_i(k) + \nu_i(k) \quad (8.2)$$

where a_i and u_i denote the acceleration and acceleration demand of vehicle i , \hat{a}_i denotes the acceleration output of the nominal model, θ_{i1}^* and θ_{i2}^* denote constant parameters, ν_i denotes additive disturbance representing actuator uncertainties. The

spacing error of the i th follower vehicle and relative speed of vehicle i and $i - 1$ are defined by

$$e_i(k) = x_i(k) + L_i - x_{i-1}(k) \quad (8.3)$$

$$\delta_i(k) = v_i(k) - v_{i-1}(k) \quad (8.4)$$

where L_i denotes the desired intervehicular space. Without loss in generality L_i can be assumed to be zero in the analysis. The position and forward speed of the i th vehicle are denoted by x_i and v_i , respectively. By using Euler approximation of integrators,

$$e_i(k+1) = e_i(k) + T_s \delta_i(k) \quad (8.5)$$

$$\delta_i(k+1) = \delta_i(k) + T_s (a_i(k) - a_{i-1}(k)) \quad (8.6)$$

the spacing error dynamics can be written for each follower vehicle as follows

$$\begin{bmatrix} e_i(k+1) \\ \delta_i(k+1) \\ \hat{a}_i(k+1) \end{bmatrix} = \begin{bmatrix} 1 & T_s & 0 \\ 0 & 1 & T_s \\ 0 & 0 & \theta_{i1}^* \end{bmatrix} \begin{bmatrix} e_i(k) \\ \delta_i(k) \\ a_i(k) \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0 \\ -T_s & 0 & -T_s & T_s \\ 0 & \theta_{i2}^* & 0 & 0 \end{bmatrix} \begin{bmatrix} a_{i-1}(k) \\ u_i(k) \\ v_{i-1}(k) \\ v_i(k) \end{bmatrix} \quad (8.7)$$

The open-loop model of the entire platoon

$$x(k+1) = Ax(k) + Bu(k) + B_\nu \nu(k) + E_d r(k)$$

is constructed by introducing the state vector $x^T = [\hat{a}_0 \ e_1 \ \delta_1 \ \hat{a}_1 \ \cdots \ e_n \ \delta_n \ \hat{a}_n]$, control input vector $u^T = [u_1 \ \cdots \ u_n]$, disturbance vector $\nu^T = [\nu_0 \ \cdots \ \nu_n]$ and reference signal $r = u_0$.

The platoon controller is a modified version of the constant spacing strategy presented in [14, Sect. 3.3.4]. The modification resides in that, instead of measured acceleration, control input is transmitted through the network. Consequently, the gear change has lower impact in the control signal than in the acceleration, so each vehicle can change gear without deceiving the followers; the vehicles react quicker to maneuver changes; and no need for filtering the rather noisy acceleration measurements. The control strategy in a general form is defined by the following equations

$$u(k) := u_L(k) + \hat{u}_N(k) \quad (8.8)$$

$$u_L(k) = K_L x(k) \quad (8.9)$$

$$u_N(k) = K_N x(k) + G_N r(k) + S u(k) + H_N \nu(k) \quad (8.10)$$

where u_L contains the locally available radar information. Gain matrix K_L can be constructed based on the following definition

$$u_{L,1}(k) = -k_1\delta_1(k) - k_2e_1(k) \quad (8.11)$$

$$u_{L,i}(k) = -k_{1\beta}\delta_i(k) - k_{2\beta}e_i(k), \quad i > 1 \quad (8.12)$$

Control signal u_N is constructed from the information received from the communication network

$$u_{N,1}(k) = u_0(k) \quad (8.13)$$

$$u_{N,i}(k) = \frac{1}{1+q_3}u_{i-1}(k) + \frac{q_3}{1+q_3}u_0(k) - k_{1\alpha}\sum_{j=0}^i\delta_j(k) - k_{2\alpha}\sum_{j=0}^ie_j(k), \quad i > 1 \quad (8.14)$$

where $k_1, k_2, k_{1\alpha}, k_{2\alpha}, k_{1\beta}$ and $k_{2\beta}$ are design parameters, see [10] for a possible choice. Matrices K_N, G_N, H_N and S can be constructed based on (8.11)–(8.14).

The communication network has a sampling time of $T = NT_s$ and the packet is transmitted after $h < T$ constant delay. If $u_N(k)$ denotes the variable to be transmitted at the network input, then

$$\hat{u}_N(k) = \begin{cases} u_N(k-h) & \text{if } \frac{k-h}{N} \text{ is an integer} \\ \hat{u}_N(k-1) & \text{otherwise} \end{cases} \quad (8.15)$$

denotes the network output at the receiver.

The closed-loop system with the delayed communication is derived in [10]. The local part u_L of the controllers run with the faster sampling rate T_s . By closing the loop with u_L , re-sampling with NT_s , then closing the loop with \hat{u}_N and assuming $r(k) = r(k+1) = \dots = r(k+N-1)$ and $\nu(k) = \nu(k+1) = \dots = \nu(k+N-1)$ we arrive at the following closed-loop model with augmented state vector

$$z(k+N) = A_z z(k) + B_{\nu,z} \nu(k) + E_z r(k), \quad z(k) = \begin{bmatrix} x(k) \\ u_N(k-N) \end{bmatrix} \quad (8.16)$$

where

$$A_z = \begin{bmatrix} A_L^N + B_0(K_N + SK_L) & B_1 + B_0S \\ K_N + SK_L & S \end{bmatrix}, \quad E_z = \begin{bmatrix} E_N + B_0G_N \\ G_N \end{bmatrix},$$

$$B_{\nu,z} = \begin{bmatrix} B_{\nu N} + B_0H_N \\ H_N \end{bmatrix}$$

$$B_1 := \sum_{j=0}^{h-1} A_L^{N-1-j} B, \quad B_0 := \sum_{j=h}^{N-1} A_L^{N-1-j} B, \quad A_L := A + BK_L,$$

$$E_N = \sum_{j=0}^{N-1} A_L^{N-1-j} E, \quad B_{\nu N} = \sum_{j=0}^{N-1} A_L^{N-1-j} B_{\nu},$$

Notice the dependence of B_1 and B_0 on communication delay h . The spacing errors can be observed through matrixes C_i defined by $e_i(k) = C_i z(k)$, $i = 1, 2, \dots, n$.

8.3 Identification of Nominal Vehicle Models

Nominal vehicle models defined by (8.1) and (8.2) are identified in the worst-case setting. Two circumstances motivate the application of this identification approach. Both the brake system and the drive-line are functioning as unknown nonlinear, hybrid systems with many thousands of program rows organizing finite state machines. An adequate description of noise statistics is not available and only reduced order models can be considered. It seems to be reasonable to assume only strict bounds on disturbances and unmodelled dynamics. Strict bounds are also useful in the worst-case analysis of spacing error bounds. On the other hand, available performance analysis tools for model sets with unmodelled dynamics may result in conservative performance bounds. Uncertainty modelling is, therefore, confined to disturbance modelling only. The corresponding peak-to-peak system norm computation for LTI systems is sufficiently accurate.

In order to obtain a preliminary view of the amount of uncertainty in the vehicle dynamics and actuators including EBS and ECU softwares, uncertainty descriptions of several different structures are identified in the section. The first one is an ARX-type model structure with time-varying parameters. The basic concept originating in the papers [7, 8] is briefly presented in the following subsection. Then, the results are extended for obtaining minimal worst-case prediction error in Sect. 8.3.2. In the second method an output error (OE) model structure is identified in Sect. 8.3.5. Both methods are applied to the experimental data of a heavy truck. The OE model structure is also applicable for the performance analysis method presented in Sect. 8.4.2.

8.3.1 Identification of the Smallest Unfalsified Parameter Sets for SISO Transfer Functions

Consider the following discrete-time linear single input single output model structure

$$G(q) = \frac{\sum_{i=1}^m b_i q^{-i}}{1 + \sum_{i=1}^m a_i q^{-i}}, \quad \theta := [a_1, \dots, a_m, b_1, \dots, b_m]^T \in P_\theta(\theta^*, \varepsilon) \quad (8.17)$$

where q is the forward shift operator. Time-varying parameter vector θ is defined in the cube $P_\theta(\theta^*, \varepsilon) := \{\theta : \|W(\theta^* - \theta)\|_\infty \leq \varepsilon\}$, where the a priori given diagonal matrix $W = \text{diag}\{\frac{1}{\varepsilon_{\theta,1}}, \dots, \frac{1}{\varepsilon_{\theta,2m}}\}$ defines the shape of the cube with edges of length $2\varepsilon_{\theta,i}$. Given input output data set $\{u(k), y(k)\}_{k=1}^l$, the problem is to find the central parameter θ^* and the minimal size ε of the cube such that for every $k = m, \dots, l$ there exists a parameter $\theta \in P_\theta(\theta^*, \varepsilon)$ not invalidated by the measurements, i.e.

$$P_\theta(\theta^*, \varepsilon) \cap D_k \neq \emptyset \quad \forall k = m, \dots, l \quad (8.18)$$

where $D_k := \{\theta : y(k) = \varphi^T(k)\theta(k)\}$ and $\varphi^T(k) = [-y(k-1), \dots, -y(k-m), u(k-1), \dots, u(k-m)]$. This problem can be solved by minimizing a convex function

as follows

$$\varepsilon = \min_{\theta^*} \max_{m \leq k \leq l} \frac{|y(k) - \varphi^T(k)\theta^*|}{\|W^{-1}\varphi(k)\|_1} \quad (8.19)$$

In the following subsection the model structure is augmented by an additive disturbance term, and the worst case prediction error is minimized while an optimal shape of the parameter cube and a bound for the disturbance are determined.

8.3.2 Unfalsified ARX Model Set of Minimal Prediction Error in ℓ_∞

With the notation of the previous section we can define the following ARX type model structure, denoted by \mathcal{M}

$$\mathcal{M} = \{ y(k) = \varphi^T(k)\theta(k) + \nu(k), \theta(k) \in P_\theta(\theta^*, \varepsilon_\theta), \nu(k) \in P_\nu(\varepsilon_a), k = 1, \dots, l \} \quad (8.20)$$

where $\varepsilon_\theta = [\varepsilon_{\theta 1}, \dots, \varepsilon_{\theta 2m}]^T$, $W = \text{diag} \left(\frac{1}{\varepsilon_{\theta, 1}}, \dots, \frac{1}{\varepsilon_{\theta, 2m}} \right)$ and

$$P_\theta(\theta^*, \varepsilon_\theta) = \{ \theta : \|W(\theta^* - \theta)\|_\infty \leq 1 \}, \quad (8.21)$$

$$P_\nu(\varepsilon_a) = \{ \nu : |\nu| \leq \varepsilon_a \} \quad (8.22)$$

The shape and size of the uncertainty set characterized by ε_θ and ε_a are unknown parameters. The only information given a priori is the data set $\{u(k), y(k)\}_{k=1}^l$.

In order to characterize consistency of the model set with the data, define hyperplane D_k in the $n + 1$ dimensional extended parameter space of $p := [\theta^T \ \nu]^T$

$$D_k := \{ p : y(k) = [\varphi^T(k) \ \nu(k)]p \}$$

Let $P(\theta^*, \varepsilon_\theta, \varepsilon_a) := \{ p = [\theta^T \ \nu]^T : \theta \in P_\theta(\theta^*, \varepsilon_\theta), \nu \in P_\nu(\varepsilon_a) \}$ denote the parameter set defining model set \mathcal{M} in the extended parameter space.

Definition 1 (Consistency) *Parameter set $P(\theta^*, \varepsilon_\theta, \varepsilon_a)$ can reproduce the data if*

$$P(\theta^*, \varepsilon_\theta, \varepsilon_a) \cap D_k \neq \emptyset \quad \forall k = m, \dots, l \quad (8.23)$$

For given data $\varphi(k)$ and model set parameters θ^* , ε_θ and ε_a the output $y(k)$ that the model set can generate lies between the bounds, $\bar{y}(k)$ and $\underline{y}(k)$

$$\bar{y}(k) := \max_{\theta \in P_\theta(\theta^*, \varepsilon_\theta)} \varphi^T(k)\theta + \varepsilon_a \leq y(k) \leq \underline{y}(k) := \min_{\theta \in P_\theta(\theta^*, \varepsilon_\theta)} \varphi^T(k)\theta - \varepsilon_a \quad (8.24)$$

With these bounds, the parameter set identification problem can be formulated as follows.

Problem 1 Assume that a data set $\{u(k), y(k)\}_{k=1}^l$ is given. Find a model set characterized by θ^* , ε_θ and ε_a such that (8.23) is satisfied and that minimizes $\gamma := \frac{1}{2} \|\bar{y}(k) - \underline{y}(k)\|_\infty$.

8.3.3 Solution via Linear Programming

It will be shown that Problem 1 leads to the solution of a linear programming (LP) problem. In contrast to the solution of [8], where for each D_k a minimum necessary size parameter $\varepsilon = \varepsilon(D_k, \theta^*)$ is determined for a given θ^* , we characterize consistency with the help of the output bounds

Lemma 1 Consistency condition (8.23) is satisfied if and only if there exist θ^* , ε_θ and ε_a such that

$$y(k) \leq \varphi^T(k)\theta^* + |\varphi^T(k)|\varepsilon_\theta + \varepsilon_a, \quad k = m, \dots, l \quad (8.25)$$

$$y(k) \geq \varphi^T(k)\theta^* - |\varphi^T(k)|\varepsilon_\theta - \varepsilon_a, \quad k = m, \dots, l \quad (8.26)$$

where $|\cdot|$ element-wise takes the absolute value of the argument.

Proof We only need to show that $\max_{\theta \in P_\theta(\theta^*, \varepsilon_\theta)} \varphi^T(k)\theta = \varphi^T(k)\theta^* + |\varphi^T(k)|\varepsilon_\theta$ and $\min_{\theta \in P_\theta(\theta^*, \varepsilon_\theta)} \varphi^T(k)\theta = \varphi^T(k)\theta^* - |\varphi^T(k)|\varepsilon_\theta$, then the statement follows from the definitions. The linear function $\varphi^T(k)\theta$ over a convex polytope takes up its extreme values at the vertices of the polytope. Let the vertex set of $P_\theta(\theta^*, \varepsilon_\theta)$ be denoted by \mathcal{V} ,

$$\mathcal{V} = \left\{ \theta : \theta = \theta^* + \begin{bmatrix} \pm\varepsilon_{\theta,1} \\ \vdots \\ \pm\varepsilon_{\theta,2m} \end{bmatrix} \right\}$$

where \pm means all combinations. From this the claims follow.

The following theorem summarizes our results.

Theorem 1 The model set \mathcal{M} which is consistent with the data set $\{u(k), y(k)\}_{k=1}^l$ and minimizes $\gamma = \frac{1}{2} \|\bar{y}(k) - \underline{y}(k)\|_\infty$ is the solution of the following LP problem.

$$\min_{\theta^*, \varepsilon_\theta, \varepsilon_a} \gamma \quad \text{subject to (25), (26) and} \quad \gamma \geq |\varphi^T(k)|\varepsilon_\theta + \varepsilon_a, \quad k = m, \dots, l \quad (8.27)$$

The problem involves $4m + 2$ variables and $3(l - m + 1)$ inequality constraints, and can be efficiently solved by rutin CLP in the MPT toolbox for Matlab, [3].

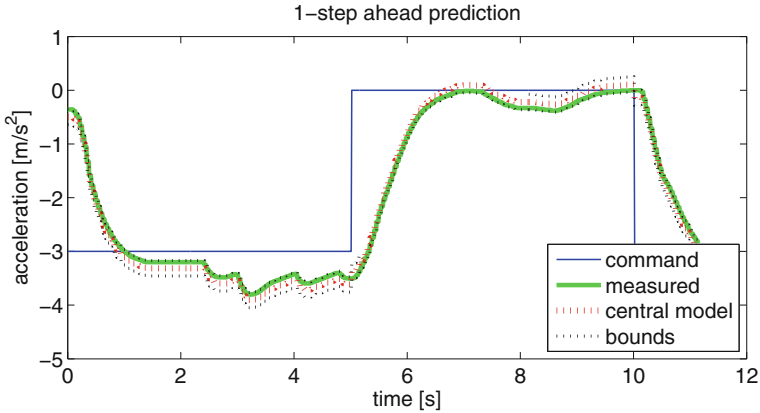


Fig. 8.1 One step ahead prediction with the central model with parameter θ^* in a braking experiment. Bounds for the prediction, \bar{y} and \underline{y} , are also plotted (*thin dotted black lines*)

8.3.4 Identification of ARX Vehicle Models

Several braking experiments have been carried out with a Volvo FH, 24 ton three-axle truck. ARX models of order $m = 1$ are identified in the following.

The LP method of Theorem 1 is applied to the model structure

$$a(k) = a(k-1)\theta_1(k) + u(k-1)\theta_2(k) + \nu(k) \quad (8.28)$$

where $\theta(k) := [\theta_1(k) \ \theta_2(k)]^T \in P_\theta(\theta^*, \varepsilon_\theta)$, $\|\nu(k) - \nu^*\|_\infty \leq \varepsilon_a$, and $a(k)$ denotes the longitudinal acceleration and $u(k)$ denotes the acceleration demand. An offset error of the measurements can be taken into consideration with parameter ν^* . The unknown parameters of the model are the central parameters θ^* and ν^* , and the bounds of the parameter and noise variation, ε_θ and ε_a , respectively.

The one-step ahead prediction of the optimal model is plotted in Fig. 8.1. The central parameters θ_1^* and θ_2^* correspond to a time constant of 1.13 s and a gain of 9.5 when the model is transformed to continuous time by zero order hold ($T_s = 0.01$ s). For the parameter variation $\varepsilon_\theta^T = [0.18 \ 0.20] \cdot 10^{-12}$ is obtained.

By fixing the maximum allowed noise level ε_a , the optimization can be performed in the remaining variables. Figures 8.2 and 8.3 show the dependence of the prediction error bound and the optimal parameters on the chosen noise levels, respectively. It can be seen that forcing the model set to represent uncertainty by the time-variation of parameters will result in overly conservative models. At the optimum, the uncertainty is described almost entirely by the noise term. A more sophisticated uncertainty description is necessary which will be provided in Sect. 8.4.2.

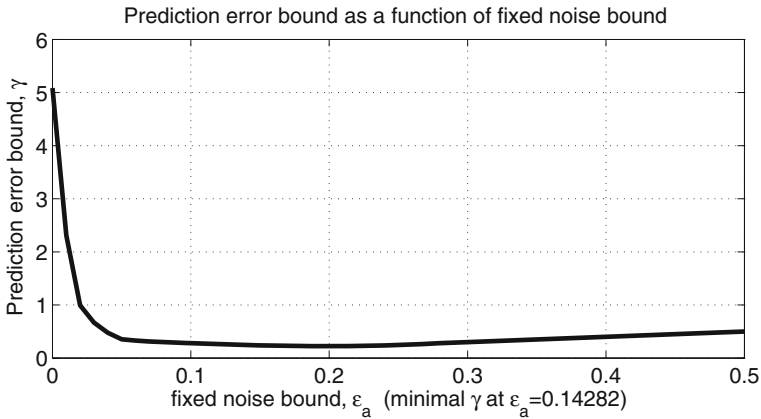


Fig. 8.2 Worst case prediction error as a function of fixed noise bound ϵ_a in the ARX model structure

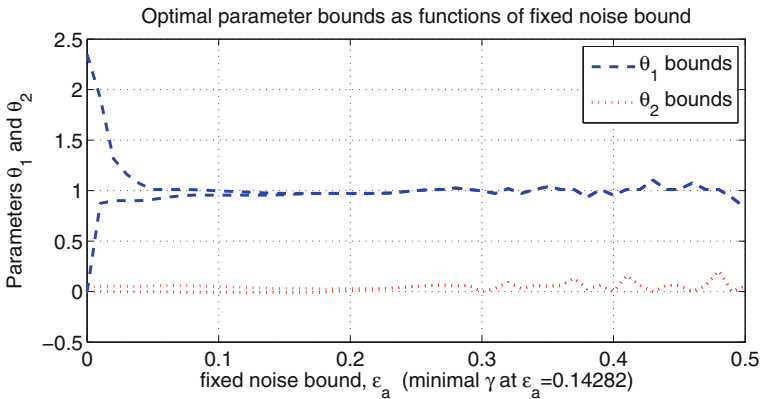


Fig. 8.3 Parameter bounds as functions of fixed noise bound ϵ_a in the ARX model structure

8.3.5 Identification of OE Models of Minimal Error in ℓ_∞

In this section an output error model structure is identified with the smallest error in ℓ_∞ . Suppose, we are given a data set $\{u(k), y(k)\}_{k=1}^l$ and the model structure of LTI SISO systems in the form

$$\hat{y}(k) = G(q)u(k), \quad G(q) = \frac{\sum_{i=1}^m b_i q^{-i}}{1 + \sum_{i=1}^m a_i q^{-i}} \tag{8.29}$$

$$y(k) = \hat{y}(k) + \nu(k) \tag{8.30}$$

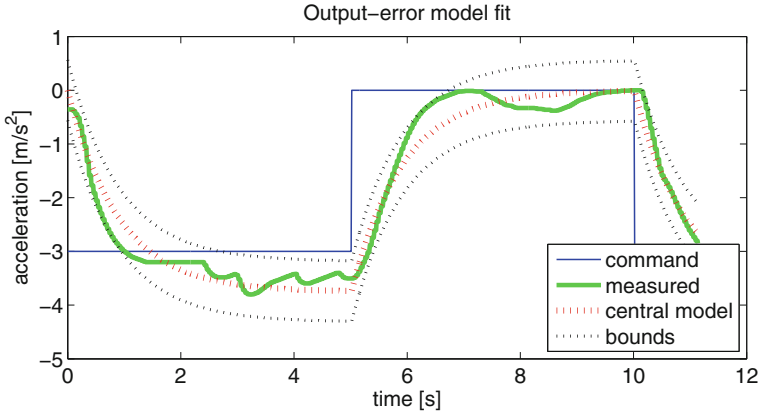


Fig. 8.4 Fit of the OE model with parameter to the measurements in a braking experiment. Bounds for the error are also plotted (*thin dotted black lines*)

The set of parameters is divided as $\theta_a = [a_1, \dots, a_m]$ and $\theta_b = [b_1, \dots, b_m]$. We are looking for θ_a and θ_b that minimize $\gamma := \|y(k) - \hat{y}(k)\|_\infty$. This optimization problem is nonlinear in parameter θ_a , therefore a nonlinear programming method can be applied. In case of small noises, good initialization for θ_a and determination of the model order can be attained by using the recent result [13]. Once θ_a is fixed, θ_b can be computed by linear programming as follows.

1. Simulate $y_j(k) = \frac{q^{-i}}{1 + \sum_{i=1}^m a_i q^{-i}} u(k)$, $i = 1, \dots, m$ and let $Y(k) = [y_1(k), \dots, y_m(k)]^T$. From this, $\hat{y}(k) = \theta_b^T Y(k)$.
2. Solve the LP problem

$$\min_{\theta_b} \gamma \quad \text{s.t.} \quad -\gamma \leq y(k) - \theta_b^T Y(k) \leq \gamma, \quad k = m, \dots, l \quad (8.31)$$

8.3.6 Identification of OE Vehicle Models

Experimental data used in Sect. 8.3.4 is applied now for identification of the OE model structure

$$a(k) = a(k-1)\theta_1 + u(k-1)\theta_2 + \nu(k) - \nu(k-1)\theta_1, \quad \|\nu(k) - \nu^*\|_\infty \leq \varepsilon_a \quad (8.32)$$

The LP method presented in Sect. 8.3.5 is applied for identifying θ_2 , while θ_1 is determined by simple line search. The optimal parameters correspond to a time constant of 0.9s and a gain of 1.25 when the model is transformed to continuous time by zero order hold ($T_s = 0.01$). The fit of the model and the error bounds are plotted in Fig. 8.4. This model can serve as nominal models in the performance analysis of the platoon.

8.4 Performance Analysis

8.4.1 Effects of Platoon Heterogeneity

For the case of heterogeneous platoons with nominal LTI models, ℓ_∞ -bounds on spacing errors are analyzed. Assume that the allowable reference input $r = u_0$ satisfies $\|u_0\|_\infty \leq u_{max}$, where u_{max} is a given bound and there are no actuator uncertainties, $\nu_i = 0$. Then, the worst-case peaks of the spacing errors, as functions of communication delays, can be computed as follows

$$\varepsilon_i := \|e_i\|_\infty = \sum_{j=0}^{\infty} |C_i A_z^j E_z| u_{max}, \quad i = 1, \dots, n \quad (8.33)$$

In the following numerical analysis ε_i , $i = 1, \dots, n$, are computed when the platoon is not homogeneous in nominal vehicle parameters θ_i^* . It is assumed that both $\theta_{i,1}^*$ and $\theta_{i,2}^*$ may differ from vehicle to vehicle

$$\Theta_{\tau g} := [\theta_{1,1}^* \theta_{1,2}^* \theta_{2,1}^* \theta_{2,2}^* \cdots \theta_{n,1}^* \theta_{n,2}^*], \quad \theta_{i,1}^* = 1 - \frac{T_s}{\tau_i}, \quad \theta_{i,2}^* = \frac{T_s g_i}{\tau_i}, \quad (8.34)$$

$$\tau_i \in \{0.6, 0.8\}, \quad g_i \in \{0.9, 1.1\}$$

where time constant τ_i and gain g_i are parameters of the continuous-time vehicle models and may take up their extremal values. It can be shown that the worst-case platoon configuration is the case when the vehicle model parameters are extremal and alternating in order. This means that if the platoon is of length $n + 1$, it is enough to compute (8.33) for 4^{n+1} systems. Taking the maximum and minimum for the 4^{n+1} systems, Fig. 8.5 shows the worst-case and best-case bounds as functions of the vehicle index i for $d_{max} = 2 \text{ m/s}^2$. The lower bounds are achieved in case of homogeneous platoons. Upper bounds correspond to platoons of alternating vehicle dynamics. For a given set of allowable maneuvers, this analysis directly provides hints on choosing safety gaps between the vehicles in the different control modes, such as $L_i > \varepsilon_i$, assuming zero initial conditions. The analysis is carried out for a range of network delays from $h = 0$ to $h = 8T_s$, but network delay of this range has negligible impact on the bounds.

In the case when gain coefficients are estimated on-line, for example with the help of parameter adaptation methods described in [14], acceleration demand can always be scaled so that θ_{i2} parameters can be set to $g_i = 1$. Then, for the uncertainty set characterized by

$$\Theta_\tau := [\theta_{1,1}^* \theta_{2,1}^* \cdots \theta_{n,1}^*], \quad \theta_{i,1}^* = 1 - \frac{T_s}{\tau_i}, \quad \theta_{i,2}^* = \frac{T_s}{\tau_i}, \quad \tau_i \in \{0.6, 0.8\} \quad (8.35)$$

the spacing errors are bounded as shown in Fig. 8.6. The bounds reduced to about one meter.

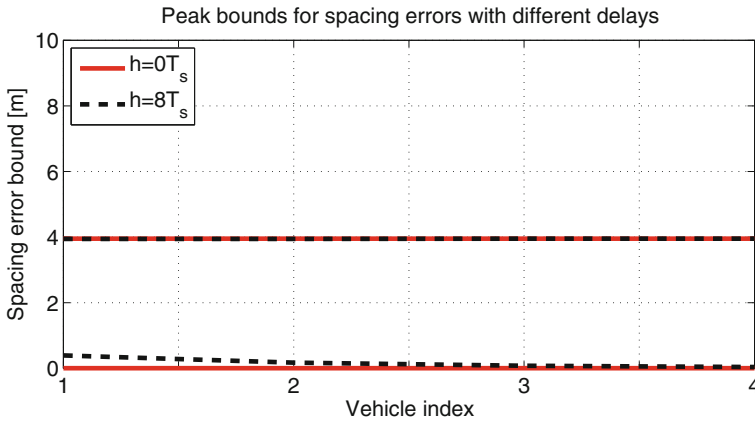


Fig. 8.5 Lower and upper bounds on spacing errors, ε_i for different network delays. Uncertainty is defined by (8.34). Lower bounds (around zero) correspond to homogeneous platoons

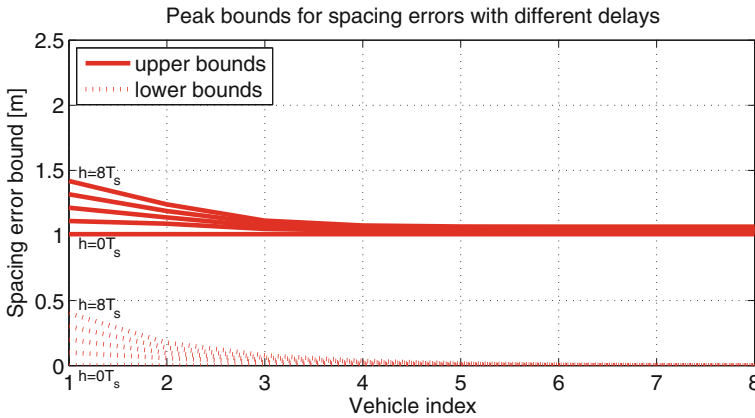


Fig. 8.6 Lower and upper bounds on spacing errors, ε_i for different network delays. Uncertainty is defined by (8.35)

8.4.2 Effects of Actuator Uncertainties

In this section, homogeneous platoons are assumed and merely the effects of brake actuator uncertainties are estimated. The appropriate contribution to the spacing errors is defined by

$$\varepsilon_{\nu, i} := \sum_{j=0}^{\infty} \sum_{l=0}^n |C_i A_z^j B_{\nu, z, l}| \nu_{l, max} \tag{8.36}$$

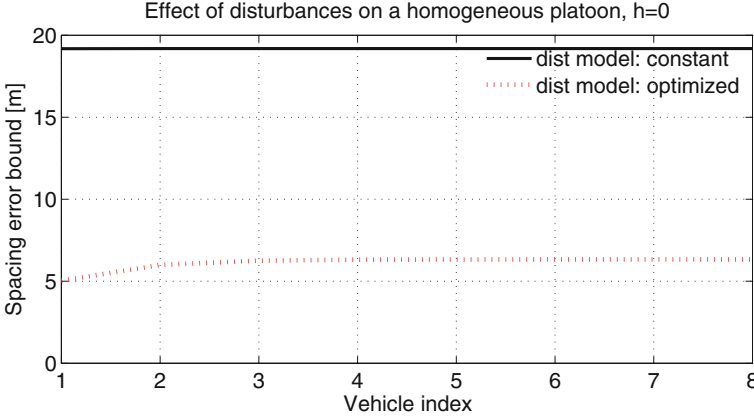


Fig. 8.7 Spacing error bounds $\varepsilon_{\nu i}$ in braking experiments

where the allowable disturbances satisfy $\|\nu_i\|_\infty \leq \nu_{i,max}$, $i = 0, \dots, n$ and $B_{\nu,z,l}$ denotes column l of $B_{\nu,z}$. It can be shown that the general case can be approximated by the sum of bounds $\varepsilon_{\nu,i}$ and ε_i obtained in this and the previous sections, respectively. For driving experiments, the case is a bit more complicated, see [12].

For some vehicle $\nu_j \leq \gamma_j = 1.13$ is obtained by the method presented in Sect. 8.3.6 for the identified ℓ_∞ -bound on the output of the nominal model. By assuming that the same bound holds for every vehicle, (8.36) is calculated for $i = 0, \dots, n$. Figure 8.7 presents, with black solid line, the calculated spacing error bounds corresponding to this disturbance model. The bound about 19 m indicates that an amplitude bounded but otherwise arbitrary additive disturbance might be a too conservative model for evaluating spacing performance. Assume, therefore, that brake actuator disturbance is generated by the model

$$\nu_i(k) = W_{\nu i}(q)\xi_i(k), \quad \|\xi_i\|_\infty \leq 1, \quad i = 0, \dots, n \quad (8.37)$$

where $W_{\nu i}$ is a bounded, stable and stable invertible operator satisfying

$$\|W_{\nu i}^{-1}(q)(a_i(k) - V_i(q)u_i(k))\|_\infty \leq 1, \quad (8.38)$$

i.e. a consistency condition with available experimental data $\{a_i(k), u_i(k)\}_{k=0}^N$. A subset of all consistent models can be finitely parameterized, for example, via finite impulse response representation, by using Laguerre or Kautz bases or by pole-zero-gain parametrization of fixed order. Let $\theta_{\nu i}$ denote the parameter vector of model $W_{\nu i}(q, \theta_{\nu i})$. Then, performance of the platoon, $\varepsilon_\nu := \sum_{i=1}^n \varepsilon_{\nu,i}$, not falsified by measurement data can be obtained as the solution of the following optimization problem



Fig. 8.8 Experimental vehicles in project TruckDAS

$$\varepsilon_\nu := \inf_{\theta_{\nu_i}, i=0, \dots, n} \sum_{i=1}^n \sum_{l=0}^n \|P_{\nu,il}(q)W_{\nu l}(q, \theta_{\nu_l})\|_1 \quad \text{s.t. (38)} \quad (8.39)$$

where $P_{\nu,il}(q)$ denotes the transfer function from disturbance ν_l to spacing error i . By using a pole-zero-gain parametrization for $W_{\nu i}(q, \theta_{\nu_i})$ with two real and a complex pair of poles and zeros, respectively, confined to a stable sector of the unit disc, the optimization provided a significant reduction of spacing error bounds to 6m, see red dotted line in Fig. 8.7.

8.5 Experimental Results

The control strategy presented in Sect. 8.2 is implemented on a platoon of three heavy trucks and tested on a 3 km long runway. The leader vehicle, driven by a driver, is a 18 ton MAN TGA two-axle tractor with load cage. The second vehicle is a 24 ton Volvo FH three-axle truck. The third one is a 18 ton Renault Magnum two-axle tractor with a semitrailer, see Fig. 8.8. All vehicles are equipped with automatic gear change. The communication network consists of radio transceivers operating on the open 868 MHz ISM narrow-band.

The experimental scenario is started with a ‘joining in’ maneuver in which the leader vehicle passes the others which are travelling at constant speed. When the last vehicle in the platoon is caught by the radar of the joining vehicle and its driver enables autonomous mode, the joining vehicle is accelerated and braked by given constant values and for sufficient time so that the vehicle arrives approximately at the prescribed distance and speed close to that of the platoon. After the braking period the spacing controller is switched on. When both joining maneuvers are finished, the leader vehicle can freely accelerate and decelerate.

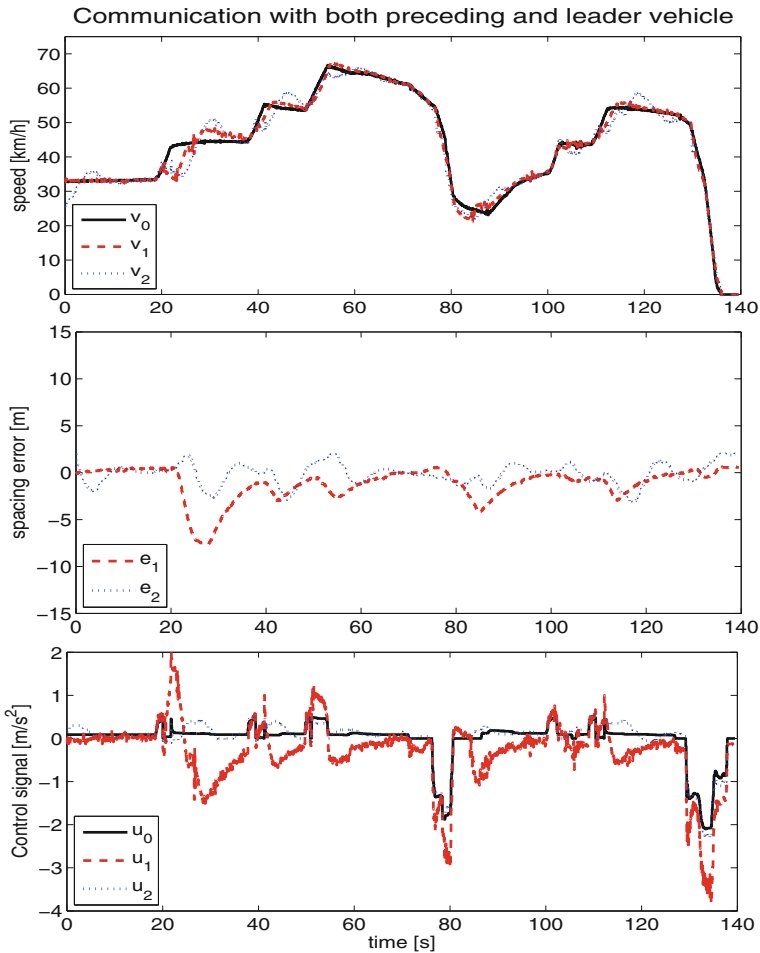


Fig. 8.9 Platoon control experiment

Nine experiments of similar maneuvers were carried out on a 3 km long road. One of them is shown in Fig. 8.9. The maximum spacing error was not greater than 3 m during braking maneuvers. During driving maneuvers, the maximum leg was not greater than 8 m.

8.6 Conclusions

Spacing error analysis of heterogeneous platoons with inter-vehicle communication and actuator uncertainties has been presented. The acceleration and deceleration commands provided by the implemented controller have been carried out by the

external demand services of ECU and EBS, respectively. According to our experiences in both unfalsification based model analysis and experimental tests with a platoon of three vehicles with different types and properties, we can conclude that a safety gap of 8 m can be safe if the acceleration/deceleration of the leader vehicle is not greater than 2 m/s^2 .

Acknowledgments The research has been supported by the Hungarian National Office for Research and Technology through the project ‘Innovation of distributed driver assistance systems for a commercial vehicles platform’ (TECH_08_A2 /2-2008-0088). This research work has been supported also by Control Engineering Research Group, Hungarian Academy of Sciences at the Budapest University of Technology and Economics.

References

1. Gerdes, J.C., Hedrick, J.: Vehicle speed and spacing control via coordinated throttle and brake actuation. *Control Eng. Pract.* **5**, 1607–1614 (1997)
2. Gustafsson, T.K., Mäkilä, P.M.: Modelling of uncertain systems via linear programming. *Automatica* **32**(3), 319–335 (1996)
3. Kvasnica, M., Grieder, P., and Baotić, M. (2004). Multi-Parametric Toolbox (MPT)
4. Liang, H., Chong, K.T., No, T.S., Yi, S.: Vehicle longitudinal brake control using variable parameter sliding control. *Control Eng. Pract.* **11**, 403–411 (2003)
5. Milanese, M.: Properties of least squares estimates in set membership identification. *Automatica* **31**(2), 327–332 (1995)
6. Milanese, M., Belforte, G.: Estimation theory and uncertainly intervals evaluation in presence of unknown but bounded errors: Linear families of models and estimators. *IEEE Trans. Autom. Control* **AC-27**(2), 408–414 (1982)
7. Nagamune, R., Yamamoto, S.: Model set validation and update for time-varying siso systems. In: *Proceedings of the American Control Conference*, pp. 2361–2365. Philadelphia (1998)
8. Nagamune, R., Yamamoto, S., Kimura, H.: Identification of the smallest unfalsified model set with both parametric and unstructured uncertainty. In: *Preprints of the 11th IFAC Symposium on System Identification (SYSID97)*, vol. 1, pp. 75–80. Kitakyushu, Japan (1997)
9. Nouveliere, L., Mammar, S.: Experimental vehicle longitudinal control using a second order sliding mode technique. *Control Eng. Pract.* **15**, 943–954 (2007)
10. Rödönyi, G., Gáspár, P., Bokor, J., Aradi, S., Hankovszki, Z., Kovács, R., Palkovics, L.: Analysis and experimental verification of faulty network modes in an autonomous vehicle string. In: *20th Mediterranean Conference on Control and Automation*, pp. 747–752. Barcelona, Spain (2012)
11. Rödönyi, G., Gáspár, P., Bokor, J., Palkovics, J.: Design and analysis of an automated heavy vehicle platoon. In: *9th International Conference on Informatics in Control, ICINCO*, pp. 31–37 (2012)
12. Rödönyi, G., Gáspár, P., Bokor, J.: Unfalsified uncertainty modeling for computing tight bounds on peak spacing errors in vehicle platoons, Submitted to ACC (2013)
13. Soumelidis, A., Schipp, F., Bokor, J.: Pole structure estimation from laguerre representations using hyperbolic metrics on the unit disc. In: *50th IEEE Conference on Decision and Control and European Control Conference (CDC-ECC)*, Orlando, FL, USA (2011)
14. Swaroop, D.: String stability of interconnected systems: an application to platooning in automated highway systems. PhD dissertation (1994)

Chapter 9

Visual SLAM Based on Single Omnidirectional Views

David Valiente, Arturo Gil, Lorenzo Fernández and Óscar Reinoso

Abstract This chapter focuses on the problem of Simultaneous Localization and Mapping (SLAM) using visual information from the environment. We exploit the versatility of a single omnidirectional camera to carry out this task. Traditionally, visual SLAM approaches concentrate on the estimation of a set of visual 3D points of the environment, denoted as visual landmarks. As the number of visual landmarks increases the computation of the map becomes more complex. In this work we suggest a different representation of the environment which simplifies the computation of the map and provides a more compact representation. Particularly, the map is composed by a reduced set of omnidirectional images, denoted as views, acquired at certain poses of the environment. Each view consists of a position and orientation in the map and a set of 2D interest points extracted from the image reference frame. The information gathered by these views is stored to find corresponding points between the current view captured at the current robot pose and the views stored in the map. Once a set of corresponding points is found, a motion transformation can be computed to retrieve the position of both views. This fact allows us to estimate the current pose of the robot and build the map. Moreover, with the intention of performing a more reliable approach, we propose a new method to find correspondences since it is a troublesome issue in this framework. Its basis relies on the generation of a gaussian distribution to propagate the current error on the map to the the matching process.

D. Valiente (✉) · A. Gil · L. Fernández · O. Reinoso
Department of Systems Engineering and Automation, Miguel Hernández University,
Av. de la Universidad s/n, 03202 Elche, Alicante, Spain
e-mail: dvaliente@umh.es
<http://arvc.umh.es>

A. Gil
e-mail: arturo.gil@umh.es

L. Fernández
e-mail: l.fernandez@umh.es

O. Reinoso
e-mail: o.reinoso@umh.es

We present a series of experiments with real data to validate the ideas and the SLAM solution proposed in this work.

Keywords Visual SLAM · Omnidirectional images

9.1 Introduction

The problem of Simultaneous Localization and Mapping (SLAM) is a key point in the field of mobile robots, since a model of the environment is often required for navigation purposes. The purpose of building a map entails a complex process, since the robot needs to build a map incrementally, while, simultaneously, computing its location inside the map. The computation of a coherent map is problematic, since the sensor data is corrupted with noise that affects the simultaneous estimation of the map and the path followed by the robot.

Typically, SLAM approaches have been classified according to several facts such as, the way to estimate the representation of the map, the main algorithm to compute a solution or the kind of sensor to gather information of the environment. For example, laser range sensors have been extensively used to construct maps. In this context, maps were mostly obtained as 2D occupancy grid maps with raw laser [1], and also 2D landmark-based maps with features described from laser measurements [2].

Recently, the tendency has turned to the utilization of visual data provided by cameras to generate the map. The applications that use these sensors are generally denoted as visual SLAM. The use of visual information has several benefits that may be exploited. Vision sensors provide a huge variety of information of the scene, being less expensive than laser sensors and more efficient in terms of consumption. Inside this group there exists different alternatives of utilization. For example, stereo-based approaches in which two calibrated cameras extract relative measurements of a set of 3D visual landmarks, determined by a visual description of its appearance [3]. Other approaches present their estimation of 3D visual landmarks by using a single camera. In [4, 5] an inverse depth parametrization is carried out to initialize the coordinates of each 3D landmark since the distance to the visual landmark cannot be directly extracted with a single camera. Some other approaches [6] has also combined two omnidirectional cameras to exploit the benefits of a wider field of view like in the case of a stereo-pair sensor.

The approach presented in this chapter assumes that the mobile robot is equipped with a single omnidirectional camera. As shown in Fig. 9.1a, the optical axis of the camera points upwards, thus a rotation of the robot moving on a plane is equivalent to a shift along the columns of a panoramic image captured.

In this chapter a different representation of the environment is proposed. To date, most of the work in visual SLAM dealt with the estimation of the position of a set of visual landmarks expressed in a global reference frame [3, 4, 7, 8]. In this work, we concentrate on a different representation of the environment: the map is formed by the position and orientation of a set of views in the environment. Each view is composed by an omnidirectional image captured at a particular position,

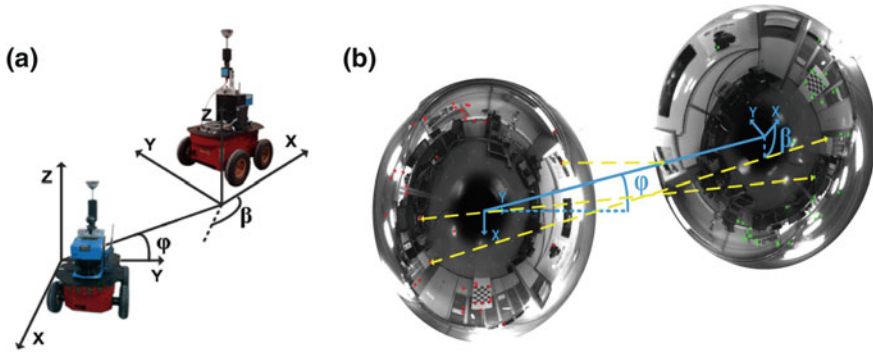


Fig. 9.1 **a** shows the sensor setup used during the experiments. **b** presents two real omnidirectional images, with some correspondences indicated and the observation variables ϕ and β

its orientation in the environment and a set of interest points extracted from it. Each interest point is accompanied by a visual descriptor that encodes the visual appearance of the point. With this information stored in the map, we show how the robot is able to build the map and localize inside it. When the robot moves in the neighbourhood of any view stored in the map and captures an image with the camera, a set of interest points will be matched between the current image and the view. Next, a set of correspondences can be found between the images. This information allows us to compute the relative movement between the images. In particular, the rotation between images can be univocally computed, as well as the translation (up to a scale factor). To obtain these measurements between images we rely on a modification of the Seven Point Algorithm [9, 10]. This idea is represented in Fig. 9.1b, where we show two omnidirectional images and some correspondences indicated. The transformation between both reference systems is also shown. The computation of the transformation relies on the existence of a set of corresponding points, thus, when the robot moves away from the position of the stored view, the appearance of the scene will vary and it may be difficult to find any corresponding points. In this case, a new view will be created at the current position of the robot. The new initialized view allows for the localization of the robot around its neighbourhood. It is worth noting that a visual landmark corresponds to a physical point, such as a corner on a wall. However, a view represents the visual information that is obtained from a particular pose in the environment. In this sense, a view is an image captured from a pose in the environment that is associated with a set of 2D points extracted from it. In the experiments we rely on the SURF features [11] for the detection and description of the points.

We consider that the map representation introduced here presents some advantages compared to previous visual SLAM approaches. The most important is the compactness of the representation of the environment. For example, in [4] an Extended Kalman Filter (EKF) is used to estimate the position of the visual landmarks, as well as the position and orientation of the camera. With this representation each visual

landmarks is represented by six variables, thus the state vector in the EKF grows rapidly as the number of visual landmarks increases. This fact poses a challenge for most existing SLAM approaches. In opposition with these, in the algorithm presented here, only the pose of a reduced set of views is estimated. Thus, each view encapsulates information of a particular area in the environment, in the form of several interest points detected in the image. Typically, as will be shown in the experiments, a single view may retain a sufficient number of interest points so that the localization in its neighbourhood can be performed.

On the other hand, we have to deal with the problem of determining a set of correspondences between two views. Determining a metric transformation between two omnidirectional images is not trivial in the presence of false data associations. In Sect. 9.3 we suggest a method to handle this issue. We introduce a gaussian propagation of the current error of the map in order to come up with a reliable scheme of matching, so that false correspondences are avoided.

We present a series of experiments and their results obtained through the acquisition of real omnidirectional images that demonstrate the validity of the approach. The set of experiments have been carried out by varying several parameters of the SLAM filter when using real images captured in an office-like environment. The chapter is organized as follows: Sect. 9.2 describes the SLAM process using the proposed framework. Next, the algorithm used to estimate the transformation between two omnidirectional images is described in Sect. 9.3. Following, Sect. 9.4 presents the experimental results. Finally, Sect. 9.5 establishes a discussion to analyze the results.

9.2 SLAM

This section describes in detail the representation of the environment as well as the map building process. As mentioned before, the visual SLAM problem is set out as the estimation of the position and orientation of a set of views. Thus, the map is formed by a set of omnidirectional images obtained from different poses in the environment. In opposition with other solutions, the views do not correspond to any physical landmarks or element in the environment (e.g. a corner, or the trunk of a tree). In our case, a landmark (renamed *view*) will be constituted by an omnidirectional image captured at the pose $x_l = (x_l, y_l, \theta_l)$ and a set of interest points extracted from that image.

This map representation can be estimated using different kind of SLAM algorithms, including online methods such as, EKF [7], Rao-Blackwellized particle filters [2] or offline algorithms, such as, for example, Stochastic Gradient Descent [12]. In this chapter we present the application of the EKF to the proposed map representation and explain how to obtain correct results using real data.

In addition we consider that the map representation and the measurement model can be also applied using standard cameras. The reason for using omnidirectional images is their ability to acquire a global view of the environment in a single image, due to their large field of view, resulting in a reduced number of variables to represent the map.

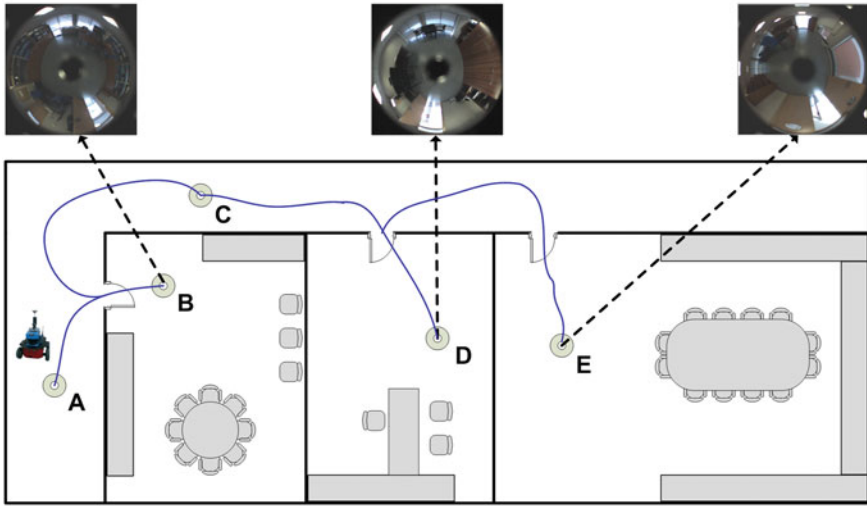


Fig. 9.2 Main idea for map building. The robot starts the exploration from A by storing a view I_A . While the robot moves and no matches are found between the current image and I_A , a new view is created at the current position of the robot, for instance in B . The process continues until the whole environment is represented

9.2.1 View-Based Map

The pose of the mobile robot at time t will be denoted as $x_v = (x_v, y_v, \theta_v)^T$. Each view $i \in [1, \dots, N]$ is constituted by its pose $x_{l_i} = (x_{l_i}, y_{l_i}, \theta_{l_i})^T$, its uncertainty P_{l_i} and a set of M interest points p_j expressed in image coordinates. Each point is associated with a visual descriptor $d_j, j = 1, \dots, M$.

Figure 9.2 describes this map representation, where the position of several views is indicated. For example, the view A is stored at the pose $x_{l_A} = (x_{l_A}, y_{l_A}, \theta_{l_A})^T$ in the map and has a set of M points detected. The view A and C allow for the localization of the robot in the corridor. The view B represents the first room, whereas the view D and E represent the second and third room respectively, and make the robot capable to localize in the environment.

Thus, the augmented state vector is defined as:

$$\bar{x} = [x_v, x_{l_1}, x_{l_2}, \dots, x_{l_N}]^T \tag{9.1}$$

where $x_v = (x_v, y_v, \theta_v)^T$ is the pose of the moving vehicle and $x_{l_N} = (x_{l_N}, y_{l_N}, \theta_{l_N})$ is the pose of the N -view that exist in the map.

9.2.2 Map Building Process

This subsection introduces an example of map building in an indoor environment, represented in Fig. 9.2. We consider that the robot explores the environment while capturing images with its omnidirectional camera. The exploration starts at the origin denoted as A , placed at the corridor. At this time, the robot captures an omnidirectional image I_A , that is stored as a view with pose x_{I_A} . We assume that, when the robot moves inside the corridor, several point correspondences can be found between I_A and the current omnidirectional image. Given this set of correspondences, the robot can be localized with respect to the view I_A . Next, the robot continues with the exploration. When it enters the first room, the appearance of the images vary significantly, thus, no matches are found between the current image and image I_A . In this case, the robot will initiate a new view named I_B at the current robot position that will be used for localization inside the room. Finally, the robot keeps moving through the corridor and goes into different rooms and creates new views respectively at these points. The number of images initiated in the map depends directly on the kind of environment. In the experiments carried out with real data we show that typically, a reduced number of views can be used while obtaining precise results in the computation of the map.

In addition, SLAM algorithms can be classified as online SLAM when they estimate the map and the pose x_v at time t . Other algorithms solve the full SLAM problem and estimate the map and the path of the robot until time t , $x_{1:t} = [x_{v_1}, x_{v_2}, \dots, x_{v_t}]$. The EKF is generally classified in the first group, since, at each time t the filter gives an estimation of the current pose x_v . However, in our case, the position of the view i coincides with the pose of the robot at any of the views. In consequence the EKF filter allows to compute the pose of the robot at time t and, in addition, a subset of the path followed by the vehicle formed by the poses $x_v, x_{I_1}, x_{I_2}, x_{I_N}$.

9.2.3 Observation Model

Following, the observation model is described. We consider that we have obtained two different omnidirectional images captured at two different poses in the environment. One of the images is stored in the map and the other is the current image captured by the robot at the pose x_v . We assume that given two images we are able to extract a set of significant points in both images and obtain a set of correspondences. Next, as will be described in Sect. 9.3, we are able to obtain the observation z_t :

$$z_t = \begin{pmatrix} \phi \\ \beta \end{pmatrix} = \begin{pmatrix} \arctan\left(\frac{y_{I_N} - y_v}{x_{I_N} - x_v}\right) - \theta_v \\ \theta_{I_N} - \theta_v \end{pmatrix} \quad (9.2)$$

where the angle ϕ is the bearing at which the view N is observed and β is the relative orientation between the images. The view N is represented by $x_{I_N} = (x_{I_N}, y_{I_N}, \theta_{I_N})$,

whereas the pose of the robot is described as $x_v = (x_v, y_v, \theta_v)$. Both measurements (ϕ, β) are represented in Fig. 9.1a.

9.2.4 Initializing New Views in the Map

We propose a method to add new views in the map when the appearance of the current view differs significantly from any other view existing in the map. A new omnidirectional image is stored in the map when the number of matches found in the neighbouring views is low. Concretely, we use the following ratio:

$$R = \frac{2m}{n_A + n_B} \quad (9.3)$$

that computes the similarity between views A and B , being m the total number of matches between A and B and n_A and n_B the number of detected points in images A and B respectively. The robot includes a new view in the map if the ratio R drops below a pre-defined threshold. To initialize a new view, the pose of the view is obtained from the current estimation of the robot pose and its uncertainty equals the uncertainty in the pose of the robot.

9.3 Computing a Transformation Between Omni-Directional Images

In this section we present the procedure to retrieve the relative angles β and ϕ between two omni-directional images, as represented in Fig. 9.1b. As shown before, these angles compose the observation described in (9.2). Computing the observation involves different problems: the detection of feature points in both images and finding a set of point correspondences between images that will be used to recover a certain camera rotation and translation. Traditional schemes, such as [13–15] apply epipolar constraints between both images and solve the problem in the general 6DOF case, whereas in our case, according to the specific motion of the robot on a plane, we reduce the problem to the estimation of 3DOF.

9.3.1 Detection of Interest Points

SURF features [11] are used to find interest points in the images and to describe their visual appearance. In [16], SURF features outperform other detectors and descriptors in terms of robustness of the detected points and invariance of the descriptor. SURF features have been previously used in the context of localization [17] using

omnidirectional images. We transform the omnidirectional images into a panoramic view in order to increase the number of valid matches between images due to the lower appearance variation obtained with this view.

9.3.2 Matching of Interest Points

In order to obtain a set of reliable correspondences between two views, several restrictions have to be considered. Some authors [10] rely on the epipolar geometry to restrict the search of correspondences. The same point detected in two images can be expressed as $p = [x, y, z]^T$ in the first camera reference system and $p' = [x', y', z']^T$ in the second camera reference system. Then, the epipolar condition establishes the relationship between two 3D points p and p' seen from different views.

$$p'^T E p = 0 \quad (9.4)$$

where the matrix E is denoted as the essential matrix which can be computed from a set of corresponding points in two images.

$$E = \begin{bmatrix} 0 & 0 & \sin(\phi) \\ 0 & 0 & -\cos(\phi) \\ \sin(\beta - \phi) & \cos(\beta - \phi) & 0 \end{bmatrix} \quad (9.5)$$

being ϕ and β the relative angles that define a planar motion between two different views, as shown in Figs. 9.1 and 9.2.

The epipolar restriction (9.4) has been previously used to compute a visual odometry from two consecutive views [10], together with some techniques such as *RANSAC* and *Histogram voting* to reject false correspondences. In this sense, the computation of the whole set of detected points is needed in order to find those which satisfy the restriction. Moreover, in the context of visual odometry, consecutive images are close enough to disregard high errors in the pose from where images were taken, so that the epipolar restriction can be normally applied. However, focusing on a SLAM framework, there exist uncertainties in the pose of the robot as well as in the pose of the views that compose the map. For this reason, we consider that it is necessary to propagate these errors to accomplish a reliable data association. We suggest using the predicted state vector extracted from the Kalman filter, from which we are able to obtain a predicted observation measurement \hat{z}_t by means of (9.2). In order to reduce the search of valid corresponding points between images, the map uncertainties have also to be taken into account, so we propagate them to (9.4) by introducing a dynamic threshold, δ . In an idealistic case, the epipolar restriction may equal to a fixed threshold, meaning that the epipolar curve between images always presents a little static deviation. On a realistic SLAM approach, we should consider that this threshold depends on the existing error on the map, which dynamically varies at each step of the SLAM algorithm. Since this error is correlated with the error on \hat{z}_t , we rename δ

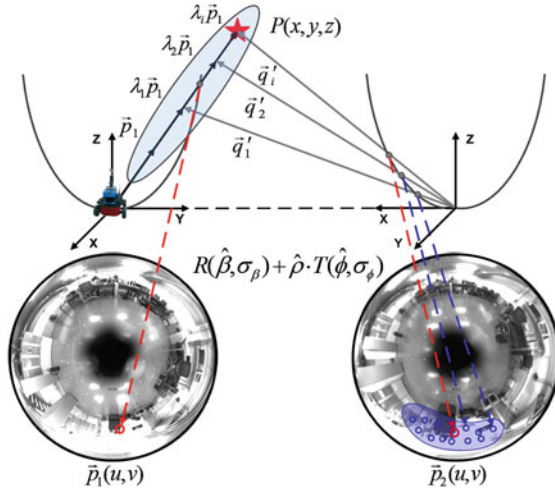


Fig. 9.3 Given a detected point p_1 in the first image reference system, a point distribution is generated to obtain a set of multi-scale points $\lambda_i p_1$. By using the Kalman prediction, they can be transformed to q'_i in the second image reference system through $R \sim N(\hat{\beta}, \sigma_\beta)$, $T \sim N(\hat{\phi}, \sigma_\phi)$ and $\hat{\rho}$. Finally q'_i are projected into the image plane to determine a restricted area where correspondences have to be found. *Circled points* represent the projection of the normal point distribution for the multi-scale points that determine this area

as $\delta(\hat{z}_t)$. In addition, it has to be noted that (9.5) is defined up to a scale factor, which is another reason to consider $\delta(\hat{z}_t)$ as a variable value. As a consequence, given two corresponding points between images, they must satisfy:

$$p'^T \hat{E} p < \delta(\hat{z}_t) \tag{9.6}$$

This approach allows us to reduce the search for corresponding points between images. Figure 9.3 depicts the procedure. Assuming a detected point $P(x, y, z)$, it may be represented in the first image reference system by a normalized vector p_1 due to the unknown scale. To deal with this scale ambiguity, we suggest a generation of a point distribution to get a set of multi-scale points $\lambda_i p_1$ for p_1 . This distribution considers a valid range for λ_i according to the predicted $\hat{\rho}$. Please note that the error of the current estimation of the map has to be propagated along the procedure. According to Kalman filter theory, the innovation is defined as the difference between the predicted \hat{z}_t and the real z_t observation measurement:

$$v_i(k + 1) = z_i(k + 1) - \hat{z}_i(k + 1|k) \tag{9.7}$$

and the covariance of the innovation:

$$S_i(k + 1) = H_i(k) P(k + 1|k) H_i^T(k) + R_i(k + 1) \tag{9.8}$$

where $H_i(k)$ relates $\bar{x}(k)$ and $z_i(k)$, $P(k+1|k)$ is a covariance matrix that expresses the uncertainty on the estimation, and $R(k)$ is the covariance of the gaussian noise introduced in the process. In addition, $S_i(k+1)$ presents the following structure:

$$S_i(k+1) = \begin{bmatrix} \sigma_\phi^2 & \sigma_{\phi\beta} \\ \sigma_{\beta\phi} & \sigma_\beta^2 \end{bmatrix} \quad (9.9)$$

Next, since the predicted \hat{E} can be decomposed in a rotation \hat{R} and a translation \hat{T} , we can transform the distribution $\lambda_i \mathbf{p}_1$ into the second image reference system, obtaining \mathbf{q}'_i . In order to propagate the error, we make use of (9.9) to redefine the transformation through normal distributions, being $R \sim \mathcal{N}(\hat{\beta}, \sigma_\beta)$ and $T \sim \mathcal{N}(\hat{\phi}, \sigma_\phi)$. This fact implies that \mathbf{q}'_i is a gaussian distribution correlated with the current map uncertainty. Once obtained \mathbf{q}'_i , they are projected into the image plane of the second image, seen as circled points in Fig. 9.3. This projection of the normal multi-scale distribution defines a predicted area in the omnidirectional image which is drawn in continuous curve line. This area establishes the specific image pixels where correspondences for \mathbf{p}_1 must be searched for. The shape of this area depends on the error of the prediction, which is directly correlated with the current uncertainty of the map estimation. Dash lines represent the possible candidate points located in the predicted area. Therefore, the problem of matching is reduced to finding the correct corresponding point for \mathbf{p}_1 from those candidates inside the predicted area by comparing their visual descriptor, instead of searching for them at the whole image.

9.3.3 Computing the Transformation

Once a set of interest SURF points have been detected and matched in two images it is necessary to retrieve the relative angles β and ϕ . In the Sect. 9.3.2 was introduced the term \hat{E} for a predicted matrix to find valid correspondences. Now this set of corresponding points is known, the real E can be determined by directly solving (9.4). The essential matrix can be expressed as a specific rotation R and a translation T (up to a scale factor), where $E = R \times T_x$. The use of a *SVD* decomposition makes able to retrieve R and T . Following [18, 19] we obtain the relative angles β and ϕ that define a planar motion between images acquired from different poses. It is worth noting that the motion is restricted to the XY plane, thus only $N = 4$ correspondences are sufficient to solve the problem. Nevertheless we use higher number of correspondences in order to get accurate solutions. Notice that now an external algorithm is not necessary to reject false correspondences since we avoid them through the restricted matching process, limited to specific image area as recently explained.

9.4 Results

We present three different experimental sets. Section 9.4.1 firstly presents SLAM results to verify the validity of this proposed approach of SLAM. In addition, it presents map building results when varying the number of views that generate the map in order to extract conclusions in the sense of the compactness of the representation. Finally, we present results of accuracy in the obtained map, use of computational resources and their variation with the number of views that conform the map. To carry out these experiments an indoor robot Pioneer P3-AT has been used, which is equipped with a firewire 1280×960 camera and a hyperbolic mirror. The optical axis of the camera is installed approximately perpendicular to the ground plane, as described in Fig. 9.1a, in consequence, a rotation of the robot corresponds to a rotation of the image with respect to its central point. Besides, in order to obtain a reference for comparison we use a SICK LMS range finder to generate a ground truth [1] which provides a resolution of 1 m in position.

9.4.1 SLAM with Real Data

This section presents SLAM results of map building to validate the proposed approach. The robot is guided through an office-like environment of 32×35 m while it acquires omnidirectional images and laser range data along the trajectory. Laser data is only used to generate a ground truth for comparison. The robot initiates the SLAM process by adding the first view of the map at the origin, as indicated in Fig. 9.4a. Next, it keeps on moving along the trajectory while capturing omnidirectional images. The image at the current robot pose is compared to the view in the map looking for corresponding points in order to extract a relative measurement of its position as explained in Sect. 9.3. The evaluation of the similarity ratio (9.3) is also computed, and in case this ratio drops below $R < 0.5$, a new view is initialized at the current robot position with an error ellipse. While the mapping continues, the current image is still being compared with the rest of the views in the map. Figure 9.4a shows the entire process where the robot finishes the navigation going back to the origin. The dash-dotted line represents the solution obtained by the proposed approach, indicating with crosses the points along the trajectory where the robot decided to initiate new views in the map and their uncertainty with error ellipses. The continue line represents the ground truth whereas the odometry is drawn with dash line. As it can be observed, a map for an environment of 32×35 m is formed by a reduced set of $N = 9$ views, thus leading to a compact representation. Figure 9.4b compares the errors for the estimated trajectory, the ground truth and the odometry at every step of the trajectory. The validity of the solution is confirmed due to the accomplishment of the convergence required by every SLAM scheme, since the solution error is inside the 2σ interval whereas the odometry error grows out of bounds.

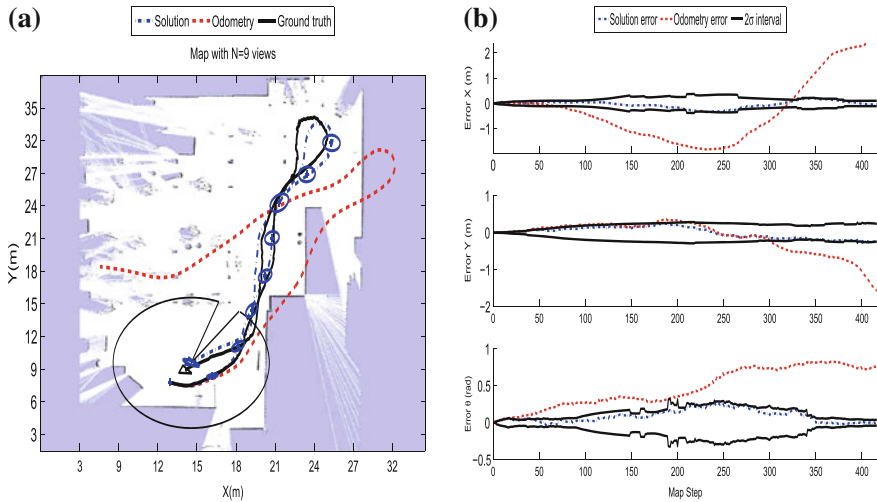


Fig. 9.4 **a** presents results of SLAM using real data. The final map is determined by $N = 9$ views, which their positions are represented with crosses and error ellipses. A laser-based occupancy map has been overlapped to compare with the real shape of the environment. **b** presents the solution and odometry error in X , Y and θ at each time step

Once the proposed approach was validated, the following experiments were carried out with the aim of testing the compactness of the representation of the environment. Another office-like environment of 42×32 m was chosen. In this case the threshold for the similarity ratio R were varied in order to get different map representation for the same environment, in terms of the number of views N that compose the map. The procedure to build up the map follows the steps detailed in the first experiment, and is depicted in Fig. 9.5. Figure 9.5a, c, e, g shows different map versions for this environment with $N = 5$, $N = 7$, $N = 12$ and $N = 20$ views respectively. Again, the estimated solution is drawn in dash-dotted line, the odometry in dash line, meanwhile the ground truth in continuous line. View's position are indicated with crosses and their uncertainty with error ellipses. Figure 9.5b, d, f, h, present the errors of the estimated solution and the odometry versus 2σ intervals to test the convergence and validity of the approach for each N -view map. All four estimations satisfy the error requirements for the convergence of the SLAM method, since the solution error is inside the 2σ interval, however the odometry tends to diverge without limit. According to this results, it should be noticed that the higher values of N the lower resulting error in the map. Nevertheless, with the lower value $N = 5$ views, the resulting error is suitable to work in a realistic SLAM problem in robotics. This fact reveals the compactness of the representation. In some context a compromise solution might be adopted when choosing between N , error and obviously computational cost. Next paragraph analysis this issue.

The concept of compactness in the representation of the map has been confirmed by the previous results. It has been observed that lower values of N provide good

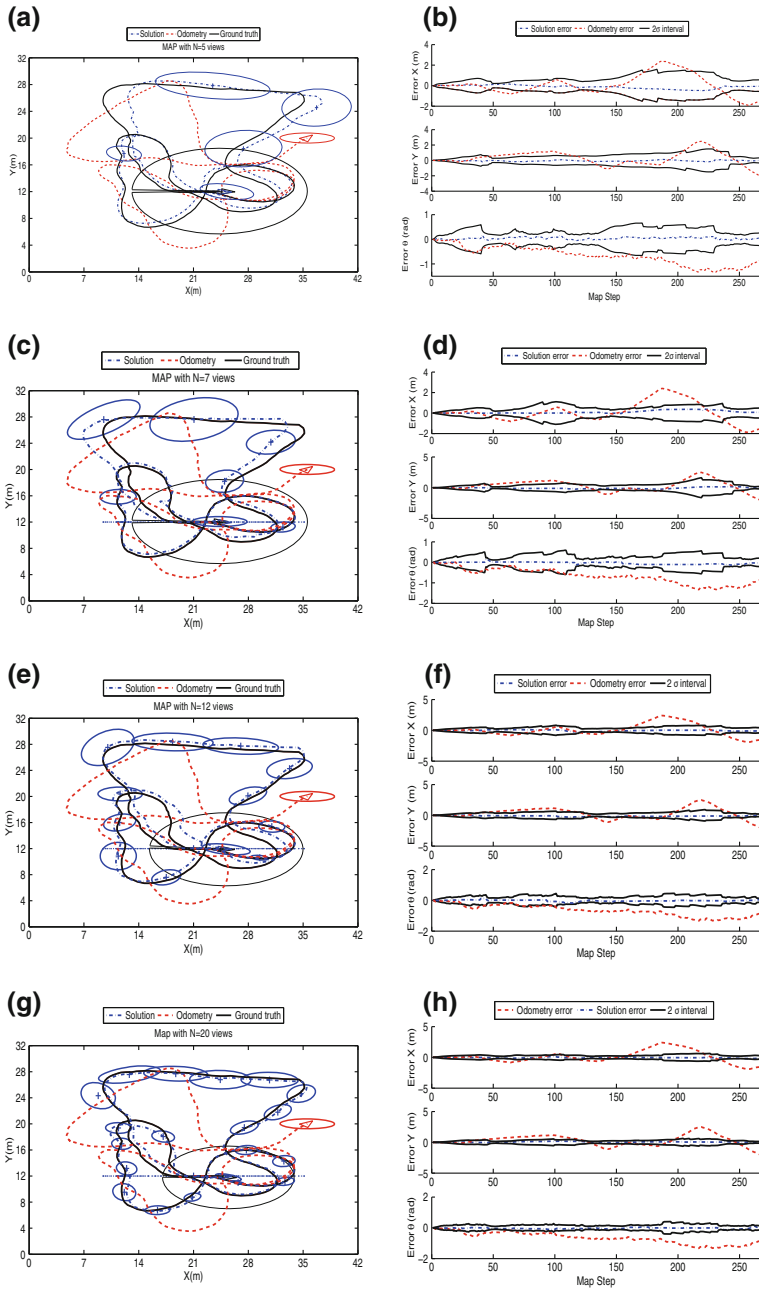


Fig. 9.5 a, c, e, g present results of SLAM using real data obtaining different map representations of the environment, formed by $N = 5$, $N = 7$, $N = 12$ and $N = 20$ respectively. The position of the views is presented with error ellipses. b, d, f, h present the solution and odometry error in X , Y and θ at each time step for each N -view map

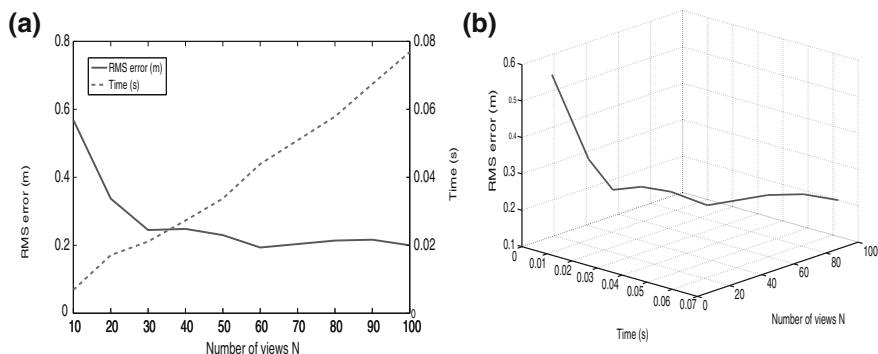


Fig. 9.6 **a** and **b** shows the RMS error (m) and time consumption (s) versus the number of views N observed by the robot to compute its pose inside different N -view maps

results in terms of error. In addition, the results allow us to analyze the computational cost and the error in terms of the number of views N composing the map. In Fig. 9.6 we present these results, showing the RMS error in position and the time consumption, which reveal that the error decreases when N increases. Consequently, the accuracy of the estimation is higher since more views are observed, however the computational cost grows. That is the reason why a compromise solution has to be reached. Generally, SLAM algorithms are real-time intended, so that the time is a limiting factor. Despite this fact, the approach presented here provides accurate results even using a reduced set of views, which is a benefit to consider when there is a lack of computational resources.

9.5 Conclusions

This work has presented an approach to the SLAM problem using a single omnidirectional camera as a visual sensor. We suggest a different representation of the environment. In contrast to traditional 3D pose estimation SLAM schemes, we only estimate the pose and orientation of a set of omnidirectional images, renamed *views*, as part of the map. A set of interest points described by visual descriptors are associated to each omnidirectional image so that a compact description of the environment is accomplished. Each omnidirectional image allows the robot to compute its localization in the image surroundings. A new matching method is suggested to deal with the problem of finding correspondences. Given two omnidirectional images and a set of interest points for each one, we model the relative error between them by means of a gaussian distribution that correlates the current estimation error on the map to help us to compute a more reliable transformation between images, composed by a rotation and a translation (up to a scale factor). This transformation allows to propose an observation model and to compute a trajectory over a map. We append

map building results using an EKF-based SLAM algorithm with real data acquisition using an indoor robot, to validate the SLAM approach. In addition we have shown the compactness of the environment representation by building maps with different number of views. Finally we presented a set of measurements to test the accuracy of the solution and the time consumption as a function of the number of views that conform the map.

Acknowledgments This work has been supported by the Spanish government through the project DPI2010-15308.

References

1. Stachniss, C., Grisetti, G., Haehnel, D., Burgard, W.: Improved rao-blackwellized mapping by adaptive sampling and active loop-closure. In: Proceedings of the SOAVE, Ilmenau, Germany (2004)
2. Montemerlo, M., Thrun, S., Koller, D., Wegbreit, B.: FastSLAM: a factored solution to the simultaneous localization and mapping problem. In: Proceedings of the 18th National Conference on Artificial Intelligence, Edmonton, Canada (2002)
3. Gil, A., Reinoso, O., Ballesta, M., Juliá, M., Payá, L.: Estimation of visual maps with a robot network equipped with vision sensors. *Sensors* **10**(5), 5209–5232 (2010)
4. Civera, J., Davison, A.J., Montiel, J.M.M.: Inverse depth parametrization for monocular slam. *IEEE Trans. Rob.* **24**(5), 932–945 (2008)
5. Joly, C., Rives, P.: Bearing-only SAM using a minimal inverse depth parametrization. In: Proceedings of ICINCO, Funchal, Madeira (Portugal) (2010)
6. Jae-Hean, K., Myung Jin, C.: Slam with omni-directional stereo vision sensor. In: Proceedings of the IROS, Las Vegas (Nevada) (2003)
7. Davison, A.J.: Murray, D.W.: Simultaneous localisation and map-building using active vision. *IEEE Trans. PAMI* **24**(7), 865–880 (2002)
8. Davison, A.J., Gonzalez Cid, Y., Kita, N.: Improving data association in vision-based SLAM. In: Proceedings of IFAC/EURON, Lisboa, Portugal (2004)
9. Scaramuzza, D., Fraundorfer, F., Siegwart, R.: Real-time monocular visual odometry for on-road vehicles with 1-point RANSAC. In: Proceedings of the ICRA, Kobe, Japan (2009)
10. Scaramuzza, D.: Performance evaluation of 1-point RANSAC visual odometry. *J. Field Rob.* **28**(5), 792–811 (2011)
11. Bay, H., Tuytelaars, T., Van Gool, L.: SURF: Speeded up robust features. In: Proceedings of the ECCV, Graz, Austria (2006)
12. Grisetti, G., Stachniss, C., Grzonka, S., Burgard, W.: A tree parameterization for efficiently computing maximum likelihood maps using gradient descent. In: Proceedings of RSS, Atlanta, Georgia (2007)
13. Kawanishi, R., Yamashita, A., Kaneko, T.: Construction of 3D environment model from an omni-directional image sequence. In: Proceedings of the Asia International Symposium on Mechatronics 2008, Sapporo, Japan (2008)
14. Nister, D.: An efficient solution to the five-point relative pose problem. In: Proceedings of the IEEE CVPR, Madison, USA (2003)
15. Stewenius, H., Engels, C., Nister, D.: Recent developments on direct relative orientation. *ISPRS J. Photogramm. Remote Sens.* **60**(4), 284–294 (2006)
16. Gil, A., Martinez-Mozos, O., Ballesta, M., Reinoso, O.: A comparative evaluation of interest point detectors and local descriptors for visual SLAM. *Mach. Vis. Appl.* **21**, 905–920(2010)
17. Murillo, A.C., Guerrero, J.J., Sagiúés, C.: SURF features for efficient robot localization with omnidirectional images. In: Proceedings of the ICRA, San Diego, USA (2007)

18. Bunschoten, R., Krose, B.: Visual odometry from an omnidirectional vision system. In: Proceedings of the ICRA, Taipei, Taiwan (2003)
19. Hartley, R., Zisserman, A.: Multiple view geometry in computer vision. Cambridge University Press, Cambridge (2004)

Chapter 10

Metrics for Path Planning of Reconfigurable Robots in Uneven Terrain

Michael Brunner, Bernd Brüggemann and Dirk Schulz

Abstract In this chapter we present metrics for rough terrain motion planning used by our hierarchical planner. We employ a two-stage planning approach which allows us to use different cost functions for an initial path search and a detailed motion planning step. To quickly find an initial path we use a roughness quantification and the operating limits of the robot, which allow a fast assessment of the drivability. We then refine the initial path in rough regions of the environment by planning the complete robot states. To determine the desired robot configurations our newly developed metrics consider the system's actuators, its safety and the time required for traversal. Real world experiments prove the validity and feasibility of the cost functions.

Keywords Metrics · Rough terrain · Reconfigurable chassis · Motion planning · Mobile robot · Autonomy

10.1 Introduction

Many robot platforms used today have a fixed chassis. Those robotic systems are limited to mainly flat environments because their design prevents them from traversing structures with sharp edges or steep inclinations. For instance, stairs, boulders, or debris are often untraversable obstacles for such platforms. However, mobile

M. Brunner (✉) · B. Brüggemann · D. Schulz
Fraunhofer Institute for Communication, Information, Processing and Ergonomics FKIE,
53343 Wachtberg, Germany
e-mail: michael.brunner@fkie.fraunhofer.de

B. Brüggemann
e-mail: bernd.brueggemann@fkie.fraunhofer.de

D. Schulz
e-mail: dirk.schulz@fkie.fraunhofer.de

Fig. 10.1 The Telemax robot is a tracked platform with four actuators. The robot is 60 cm long, 40 cm wide, and weighs about 70 kg. It has 4 tracks which can be rotated 170° from entirely folded all the way down lifting the robot about 45 cm up. Completely expanded the robot has a length of 120 cm. The robot is equipped with a skid-drive, and its maximal translational speed is 1.2 m/s



robots with reconfigurable chassis can alter their configuration, which enhances their mobility and enables them to traverse a wide variety of obstacles.

Driving a mobile robot across the above mentioned obstacles is a challenging task even for a trained operator. There are many aspects which have to be considered when driving over obstacles, most of which can be neglected for 2D navigation on flat terrain. Especially, the stability of the robot is essential as the robot falls over more easily. Inertia and momentum are increasingly important when a fast robot is operated close to its limits. Moreover, the same actuator and driving commands may lead to varying behavior of the robot caused by different contact points.

Introduced planning and controlling algorithms for traversing rough terrain or overcoming obstacles depend heavily on the robot's shape, actuators, and abilities. This may be comprehensible by considering that these systems ought to traverse very rough terrain for which they must operate at their limits. This, in turn, requires to exhaust the often unique capabilities given by their specific actuators.

We developed a two-stage motion planning approach for tracked reconfigurable robots to traverse rough outdoor environments. We generate a preliminary path using the platform's operating limits instead of the complete robot state and, subsequently, we apply a detailed motion planning step to refine the initial path in rough regions. Our algorithm is general in the sense that we do not rely on predefined motion sequences or obstacle classifications. Furthermore, our algorithm can be used for different robot models with similar locomotion like the Telemax robot (Fig. 10.1).

The metrics used by our motion planner are at the center of this chapter. Many different metrics and cost functions are proposed for path planning in 2D navigation as well as for rough terrain traversal. Some quantities, like time or path length, are applicable to both scenarios. In 2D navigation, aspects like stability, traction, or risk are usually neglected because they can safely be assumed. In contrast, quantifying these aspects during planning on rough terrain is essential. In our approach, we use a roughness quantification to adjust the robot's actions. If performed solely, actuator movements do not increase the path length; thus, we measure the path execution time

to account for these actions. In rough areas we employ to often employed quantities, e.g. stability or traction, and incorporate them as costs into our approach to determine appropriate robot configurations.

The remainder of this chapter is structured as follows: Sect. 10.2 provides an overview of some related work. We give a brief scheme of our algorithm in Sect. 10.3. Section 10.4 discusses issues of the sensor coverage for obstacle traversal, the necessity of global information, and introduces the roughness quantification. Sections 10.5 and 10.6 illustrate the metrics used for the initial path search and the metrics utilized by the detailed planner in hazardous areas, respectively. We provide real world experiments in Sect. 10.7 and conclude in Sect. 10.8.

10.2 Related Work

Extensive work has been done on traversability metrics and cost functions for path planning of mobile robots. A binary notion of traversability and the path length are generally accepted as sufficient measures in 2D navigation. However, as the terrain becomes more challenging, more detailed traversability assessments are used and further aspects, like the stability, the amount of turning, or traction, are considered during path planning. In this section we outline some of the previous work done in this area of research.

The National Institute of Standards and Technology (NIST) has introduced stepfields as a means of repeatable test methods for robot mobility to capture statistically significant performance information [1]. The NIST also proposed three metrics for stepfields: two concerning the coverability of the terrain, i.e. a difficulty measure of the entire region, and one called crossability which depicts the difficulty to move between two specified locations. While the coverability is based on the variations in height difference, the crossability is given by the least cost path in terms of the terrain roughness along the path, the amount of turning and the path length. All metrics are scaled by the robot size and wheel diameter or track height since these parameters influence the ability of the robot to overcome obstacles [2]. The NIST's crossability metric is based on [3] where the same quantities are used to include the robot model into the traversability measure. Since we intend to use the traversability measure to guide our search algorithm, a global definition like the coverability is not suitable. However, similar to the crossability we consider roughness, the amount of turning and the path length for planning. In contrast, since the robot size and wheel or track height are not fix for robots with reconfigurable chassis, we do not include these parameters. Further, we measure the execution time of a path rather than the path length to account for actuator movements, which do not influence the path length but require time.

In the area of planetary rovers most of the proposed traversability concepts are heavily based on terrain, robot and dynamic models to capture the terrain-vehicle interactions in detail [4]. Mechanical models are used to identify the cohesion and internal friction angles of the terrain in order to quantify traversability [5].

The Traversability Index is a fuzzy rule-based measure quantifying the slope of the terrain and the roughness with respect to the size and density of rocks within the camera frame [6]. It was extended in [7], additionally incorporating the terrain discontinuity (i.e. cliffs) and hardness as it affects traction. In our approach we also incorporate an estimate of the vehicle traction; however, it is less model based.

Many different stability margins were proposed for measuring the stability of a mobile robot. A comparative overview can be found in [8]. A common way to measure the static stability of a system is to project the center of mass onto the supporting polygon. Other stability margins are more accurate as they consider the height of the center of mass. In [9] the Force-Angle Stability Margin (FASM) is incorporated as slowness value into the Fast Marching Method (FMM) to favor more stable paths. As we do not contemplate any forces at this point we use the Normalized Energy Stability Margin (NESM) [10], also employed in [11].

Others classify the environment using fuzzy rules and Markov Random Fields to generate behavior maps which encode preconditions and costs for a skill-based traversal algorithm [12]. Or they label the terrain through geometric heuristic rules as flat, vertical, stairs or unknown along with associated costs (penalizing steep slopes and rough terrain) to apply specific motion primitive planners for each class [13]. However, we do not want to base our motion planning algorithm on a structure classification scheme as this will limit the algorithm to the set of defined structures. Hence, our metrics solely use general properties, rather than semantic structure labels.

10.3 Brief Algorithm Overview

In this section we give a short overview of our algorithm. We employ a two-stage planning algorithm for reconfigurable robots to traverse rough environments and to overcome obstacles in urban areas. Given a map we first compute the roughness quantification (Sect. 10.4) and build a motion graph according to the robot's operating limits (Sect. 10.5). We perform a Dijkstra search to find the initial path in this motion graph. Afterwards, we identify the path segments leading through rough terrain and construct a state graph for a tube-like area around each rough path segment (Sect. 10.6). Using a Dijkstra search we find sequences of robot states including the desired actuator positions. Finally, applying a default configuration for segments in flat areas, all segment paths are combined to provide the final path. The algorithm and its main features are introduced in [14, 15].

10.4 Map and Roughness Quantification

Whether a given structure is traversable or not is not easily determined. In 2D navigation this is usually handled with a simple threshold on the height differences; everything above this threshold is untraversable. When aiming at overcoming structures,

this question becomes very hard to answer. For 2D navigation a 2D laser range finder is sufficient to gather the necessary information about the surroundings. In contrast, using a 3D sensor for traversing obstacles and navigating through rough environments is not enough, due to the still very limited sensor coverage.

First, installing sensors on a mobile robot introduces a very narrow view; second, determining what will come after successfully climbing onto an obstacle is very difficult when using common sensors; and third, while traversing an obstacle the robot's pose will often orient the sensors in a way that they are unable to cover the environment. Consider, for example, the traversal of a flight of stairs; the very narrow view makes it hard to recognize the stairs especially all the way up; and while on the stairs and close to the top the sensors cover very few of the ground. This means, deciding on the traversability of an obstacle based on local sensor information is very hard. Therefore, we use a map to simplify the perceptual task and to be able to concentrate on the planning aspect of the problem of rough terrain traversal. Means of how to deduce the traversability of a greater area from local sensor information is beyond the scope of this work.

Further, discovering that an obstacle is actually untraversable during traversal is an unpleasant situation. In this case the robot may be on top of an obstacle and be required to retreat backwards or, worse, since obstacles are generally not traversable from every angle, the robot may be stuck and, thus, unable to move safely off the obstacle. A map may help reducing such incidents but certainly cannot account for all situations, as a map usually is not detailed enough and physical aspects, like friction coefficients, are generally unavailable.

Also related to the first point, especially for rough environments it cannot be assumed that a path to a desired goal exists at all. Moreover, a map allows to assess the risk of a path and, thus, enables to determine whether driving through a hazardous area is worth the risk or circumventing the obstacle with reasonable additional costs is more appropriate.

In addition, a map may be used to improve localization by comparing the robot's actual pose with the expected pose deduced from the map. However, the localization and an enhanced controller are subject to future work.

Using a map for motion planning in rugged environments introduces another problem. The validity of the planning is closely related to the level of detail of the map. However, large detailed maps are rarely available. This can be addressed by detailed patches for rough regions and a greater coarse map.

10.4.1 Roughness Quantification

In our approach we use a height map to represent the environment because it is simple to use and sufficient for our application. In order to assess the difficulty of a position within the map, we use techniques from image processing to compute a roughness quantification of the map (Fig. 10.2). As indicator how difficult an area of the map is, we use the height differences between cells. For a given map cell $c(x, y)$

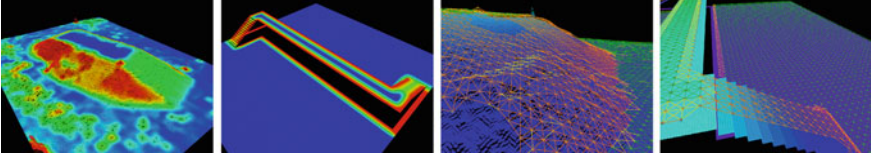


Fig. 10.2 The *left* images show the roughness quantification of two maps; a real map build by a laser range finder and an artificial urban map. The *colors* indicate the degree of roughness, ranging from *dark grey* for flat regions to *white*, high risk areas. The *right* images show the motion graphs for two maps; a map of a training hill and an artificial urban map

we determine the maximal height difference within a window w of size $k_x \times k_y$ around the cell, i.e., we apply a maximum filter

$$c(x, y) = \max_{(i_1, j_1), (i_2, j_2) \in w_{x,y}} \{|c(i_1, j_1) - c(i_2, j_2)|\}$$

with

$$w_{x,y} = \{(i, j) : |x - i| \leq k_x \wedge |y - j| \leq k_y\}.$$

However, we prevent a distortion of the range of values through a threshold h_{max} , which conveniently can be set to the robot model's maximal traversable height. Further, the values are scaled to $[0, 1]$ using h_{max} . Subsequently, we perform a Gaussian blur on this grid of height differences to smooth the transitions and, more importantly, to propagate the risk, i.e. inflate large heights and sharp edges. The Gaussian kernel for a two dimensional blur is defined as

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{\sigma^2}}.$$

Given kernel sizes of k_x and k_y for both dimensions, this leads to a matrix $G \in M(k_x + 1, k_y + 1)$ with elements

$$G(i, j) = \frac{1}{2\pi\sigma^2} e^{-\frac{\left|i - \frac{k_x}{2}\right|^2 + \left|j - \frac{k_y}{2}\right|^2}{\sigma^2}},$$

where $k_x/2$ and $k_y/2$ indicate integer divisions. For each cell in the map we convolve this kernel matrix with the window of the same size around the cell.

Note, comparing, for instance, a flight of stairs to a ramp of the same gradient angle, the steps provide less contact points for traversal with a tracked robot, and turning is much more severe on these discrete contacts. Hence, a flight of stairs should have a higher risk compared to a ramp of the same gradient angle. Using the introduced formulation allows us to distinguish between those two structures in exactly this way. Moreover, using an appropriate window size allows us to virtually inflate the hazardous areas which is commonly done in 2D navigation to keep the

robot away from obstacles. In contrast, high risk areas are avoided by the robot but, if required, do not prohibit traversal. Another benefit is that the computation is simple and highly parallelizable.

We use this roughness quantification in both metrics, for the initial path search as well as for the detailed motion planning, to adjust the behavior according to the difficulty of the environment.

10.5 Metrics for the Initial Path Search

Driving with reconfigurable robots on rough terrain and over obstacles introduces a large search space, and more constraints compared to 2D navigation must be satisfied. Additionally, the robot's actuators must be incorporated into the planning process and the quality of the path must be judged not only by its length but also by the robot's stability and traction, the amount of turning and time required for actuator movements. We, therefore, employ a first path search to quickly find an environment-driven and fast path to the desired goal location. The path is subsequently used to restrict the search space for the second planning phase which determines the final path consisting of the robot configurations including the actuator commands.

Our initial path search employs the introduced roughness quantification. This roughness quantification is used to steer the robot away from hazardous areas and to prefer less risky routes. Usually, an environment consists of fairly flat areas as well as rugged and challenging parts. In flat areas, the consideration of the robot's complete configuration including actuator values is not necessary. In contrast, it is essential in rough areas to increase robot safety and ensure successful traversal. At this planning phase, we do not know through which parts of the environment the path will lead; therefore, we omit the complete state during this planning stage. We rather stick to the operating limits of the mobile base neglecting the actuators. These operating limits do include the maximal roll and pitch angles before tipover as well as the maximal traversable height. We identify the least restrictive operating limits through the most stable configuration on flat ground. However, using a greater set of configurations generally improves the robot states in terms of stability as more configurations can compensate for a greater variety of situations.

We build a motion graph $G_m = (V_m, E_m)$ which represents the ability of the mobile robot to traverse the environment (Fig. 10.2). The vertices $v_i \in V_m$ model positions p_i of a dense¹ regular grid. A vertex v is added to V_m if the maximal risk value r within the robot's footprint does not exceed some threshold. If the transition from a position p_i to a neighboring position p_j does not violate the robot's limits in terms of inclination and height differences, the edge $e_{ij} = (v_i, v_j)$ is included in E_m . By using a graph based formulation and encoding the costs in the edge weights, we are able to perform a Dijkstra search.

¹ The resolution of the grid is usually half the robot size to avoid the requirement of intermediate validity test. Also, situations in which solutions are lost due to the discretization are few.

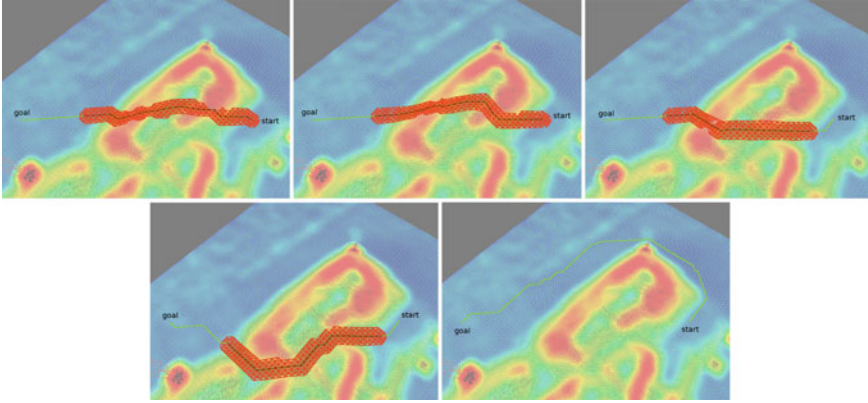


Fig. 10.3 Influence of the safety weight on the initial path search. It determines the primary direction of the final path. Start and goal position are kept the same and the safety weights set to 0.0, 0.25, 0.5, 0.75 and 1.0 for *top left* to *bottom right*. To determine the final robot configurations the second planning step considers the tube-like area around rough path segments

Using the roughness quantification, we are able to define the required time $t_v(i, j)$ to move from a position p_i to a neighboring position p_j as a simple function of the risk. Let d_{ij} be the distance between p_i and p_j , and let $r_{ij} = \frac{1}{2}(r_i + r_j)$ be the average risk of the involved positions. Then t_v is defined as

$$t_v(i, j) = \frac{d_{ij}}{\max(v_{min}, (1 - r_{ij} \cdot w_{is}) \cdot v_{max})}, \quad (10.1)$$

where v_{min} and v_{max} are the minimal and maximal forward velocity, respectively. $w_{is} \in [0, 1]$ is the safety weight for the initial path search; low values will diminish the influence of the roughness quantification and, hence, lead to possibly shorter yet more risky paths. In contrast, high values increasingly force the robot to take low risk paths. The less risky the area, the faster the robot can drive and the shorter is the time. Therefore, w_{is} determines the main direction of the final path. Figure 10.3 shows the different paths obtained for a given start and goal position and the safety weights 0.0, 0.25, 0.5, 0.75 and 1.0. For a weight of 0.0 the path becomes a direct path to the goal position. Increasing the safety weight forces the algorithm to face the direction of the steepest gradient when climbing inclinations and to avoid more and more high risk areas. Eventually, the path avoids even less risky regions and circumvents the hill.

The main purpose of this planning phase is to find a path to the goal which can be used to restrict the search space for the subsequent detailed motion planner. Therefore, we differentiate between areas with risk values and slopes below a convenient level, areas with higher risk levels and convenient slopes, and areas with higher risk values and higher slopes (see Fig. 10.2).

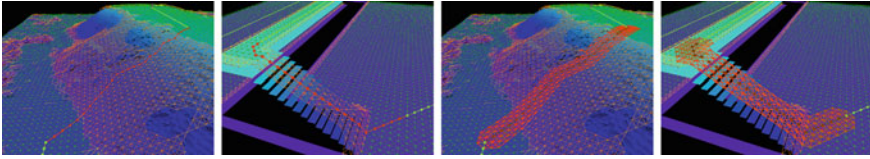


Fig. 10.4 The *left* images show the path segments of the initial paths on two maps. The *right* images show the tubes around the initial paths in rough areas

We split the path into segments leading through flat areas with low risk or no environmental slopes and segments through rough regions with high risks and slopes (left images of Fig. 10.4). For flat segments the stability of the robot system can be safely assumed as done in 2D navigation. Further, any robot configuration may be applied with no or little risk (given the pose is stable in itself). Therefore, we do not perform a detailed motion planning of the robot’s actuator configurations for flat segments. However, rough regions require an additional planning of the robot’s actuator controls and the consideration of the stability to ensure safety and task completion. This planning phase and the metrics used are described in more detail in Sect. 10.6. In regions with higher risk values but still convenient slopes we reduce the velocity of the robot.

Distinguishing between flat and rough areas during this planning stage allows us to speed up planning since we are avoiding unnecessary planning in a high dimensional space for easily accessible parts of the environment. However, we ensure the stability and safety of the robot through a detailed planning step for rough path segments. Disregarding the robot’s state and using the operating limits simplifies the traversability assessment; hence, the initial path search is fast.

10.6 Metrics for Planning in Rough Areas

Compared to flat environments, driving on rough terrain is more challenging and exposes the mobile robot to a greater risk. Therefore, using simply the operating limits of the robot is not sufficient. We need to consider the complete robot configuration including actuator values to estimate the robot’s stability and traction. The configuration of a reconfigurable robot may look like

$$(x, y, z, \theta, \psi, \phi, v, \omega, \dot{v}, \dot{\omega}, a_1, \dots, a_n),$$

where the first part describes the 6D pose of the robot. The translational and rotational velocities are v and ω , and the corresponding accelerations are depicted by \dot{v} and $\dot{\omega}$. a_i are the control values of n actuators. The stability or the contact points with the environment may also be included in this state. Reducing the state vector to the controllable part leads to

$$(x, y, \theta, v, \omega, \dot{v}, \dot{\omega}, a_1, \dots, a_n).$$

The controllable states still build a large and high dimensional search space, which cannot be searched exhaustively. Therefore, we introduce a graph-based approach using the initial path as basis to restrict the space of robot states to a tube around this path (right images of Fig. 10.4). We assume that the best path with respect to the complete robot configurations complies with or is close to the initial path. This assumption mostly holds. It would be violated primarily if time consuming actuator movements are necessary by which no distance is gained. This rarely happens since, first, we favor simultaneous execution of movements (as described later) and, second, the initial path prefers less risky routes which generally require less actuator movements. Also the best path considering the robot's entire configuration must be a fast path too.

By using a subsequent planner to refine the path in rough areas, we are able to apply another cost function. This, in turn, allows us to increase the importance of the robot's safety. The detailed motion planning accounts for the system's stability and traction and for the time consumed by rotation and actuator movements in order to prevent unnecessary actions. Since the robot's speed is very low when traversing hazardous areas, we neglect forces and dynamic stability for now. The terms of the cost function for this planning phase can be divided into a safety term and a time term.

First, we define the state graph $G_s = (V_s, E_s)$ which models a discrete subspace $X_s \subset X$ of the state space. Each vertex $v_i \in V_s$ corresponds to a state $x_i \in X_s$. Each edge $e_{ij} \in E_s$ models a valid transition from state x_i to x_j . The validity of a transition is subject to the movement constraints of the robot model. The edge weights are given by the cost function derived in this section. In the following we will use the state space notation to introduce the metrics.

10.6.1 System Safety Metric

The safety of the system is affected by several factors. We incorporate the roughness quantification, the system stability and an estimate of the traction into the safety metric. To quantify these values we approximate the robot's footprint by the best fitting plane using a least-squares method.

System Stability. We assess the static stability of the robot using the Normalized Energy Stability Margin (NESM), see [8] for a comparative overview of several stability margins. In contrast to the commonly used projection of the center of mass onto the supporting polygon, the NESM directly provides a notion of quality.

We will give a short overview of the Normalized Energy Stability Margin, for a detailed discussion see [10]. The NESM basically indicates the amount of energy which must be applied to tip the robot over the "weakest" edge of the supporting polygon. It is derived from the Energy Stability Margin (ESM) [16], see Fig. 10.5.

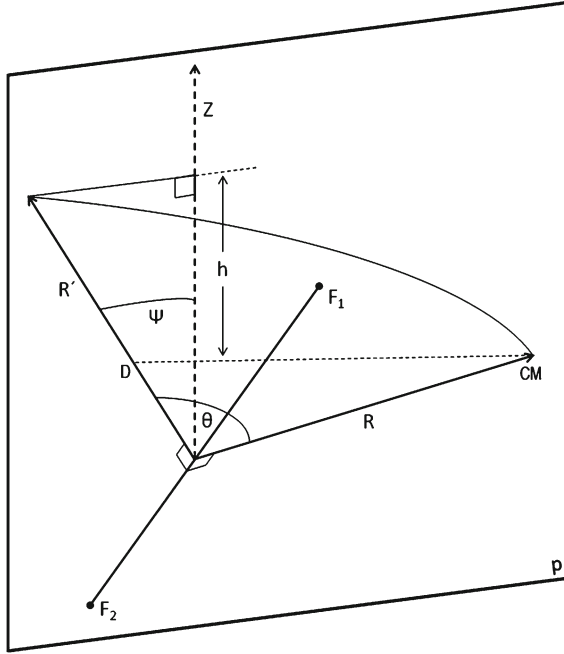


Fig. 10.5 Illustration of the Energy Stability Margin. F_1 and F_2 represent a border of the supporting polygon, i.e. a rotation axis within the plane p . R is the vector from the border to the center of mass (CM). Θ depicts the angle between R and the vertical plane and ψ the inclination of the rotation axis with respect to the horizontal plane. R' is obtained by rotating R around the rotation axis until it is contained in plane p . D is computed through $D = |R|(1 - \cos(\Theta))$ and $h = |R|(1 - \cos(\Theta)) \cdot \cos(\psi)$ provides the energy stability level. Finally, the NESM is defined as $s = \min_i(h_i)$ with h_i being the normalized energy level of the i th edge of the supporting polygon

The rotation axis is given by F_1 and F_2 as a border of the supporting polygon within the plane p . R is the vector from the border to the center of mass (CM). The angle between R and the vertical plane is given by Θ , and ψ depicts the inclination of the rotation axis with respect to the horizontal plane. R' is obtained by rotating R around the rotation axis until it is contained in plane p . D is computed through $D = |R|(1 - \cos(\Theta))$. Finally,

$$h = |R|(1 - \cos(\Theta)) \cdot \cos(\psi) \tag{10.2}$$

provides the energy stability level for the given rotation axis. The NESM is now defined as

$$s = \min_i(h_i), \tag{10.3}$$

where h_i is the normalized energy level with respect to the i th boundary of the supporting polygon. Our stability cost is the stability of a robot state x , i.e.

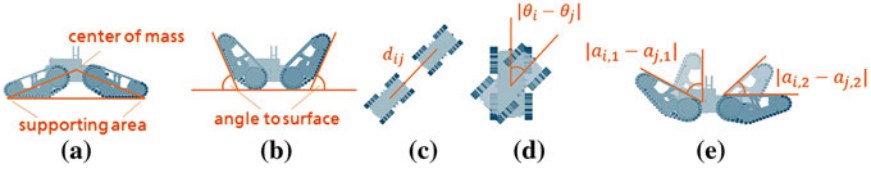


Fig. 10.6 Illustration of cost parameters. **a** Stability, **b** Traction, **c** Translation distance, **d** Rotation distance, **e** Actuator distance

$$S = 1 - \xi_s \cdot s(x), \quad (10.4)$$

where $\xi_s = \frac{1}{s_{max}}$ is a normalization term using the maximal stability value of a given robot model to scale the cost to $[0, 1]$.

The computation of the stability of the system depends on the accuracy of the center of mass. Therefore, we compute the distributed center of mass, as

$$CM = \sum_{i=1}^n c_i \cdot m_i, \quad (10.5)$$

where CM is the position of the center of mass of the system. c_i and m_i are the centers of mass and the masses of the n body parts, i.e. the chassis and the actuators. Thereby, we can determine the center of mass with respect to the actuator positions which increases the accuracy of the position of the center of mass (Fig. 10.6a).

Due to the minimum function in the stability margin the stability value may stay the same for several actuator positions. In order to reach a more stable configuration the robot might have to perform a sequence of actuator movements which do not have an immediate gain in stability. Therefore, we need another quantity which helps to pass through such sequences. We decided to use an estimate of the robot's traction.

Traction Estimate. The traction of the robot is increasingly important when traversing rough terrain or obstacles. However, we do not want to use any information about the surface properties because maps with information accurate enough to aid planning are very hard to obtain. Therefore, we estimate the actuators' ground contact as an indicator of the traction. The ground contact of an actuator a_k is defined as its angle to the surface (Fig. 10.6b). Then, the traction cost T of a state $x \in X_s$ is given as the average over those n angles.

$$T = \xi_t \cdot \frac{1}{n} \sum_{k=1}^n \psi(a_k), \quad (10.6)$$

where $\psi(a_k)$ is a function providing the angle to the surface of an actuator a_k . $\xi_t = \frac{1}{\pi/2}$ normalizes the cost to $[0, 1]$. Hence, the smaller the angle, the greater the estimated traction, the safer the robot state in terms of traction. Note, measuring the ground

contact of the actuators in terms of their angle to the surface provides a qualitative measure of the traction. It can be viewed as an indicator for traction as the traction between two objects generally increases with the contact area between those objects. Also, this estimate serves our purpose of lowering the robot's actuators on rough terrain.

The combination of the stability cost, the traction cost and the risk value of the roughness quantification constitutes the safety term c_{safety} of our cost function. Given two states $x_i, x_j \in X_s$ the safety cost is defined as

$$c_{\text{safety}}(i, j) = \frac{r_{ij} + \frac{1}{2}(S_{ij} + T_{ij})}{2}, \quad (10.7)$$

where $r_{ij} = \frac{1}{2}(r_i + r_j)$ is the average risk of both states. Similar, $S_{ij} = \frac{1}{2}(S_i + S_j)$ is the average stability cost of both involved states, and $T_{ij} = \frac{1}{2}(T_i + T_j)$ is the average traction cost.

10.6.2 Execution Time Metric

Besides a save path we also want to find a fast path to the goal. In addition to the initial path search, we consider the time required for rotation and actuator movements. The three quantities are illustrated in Fig. 10.6c–e. For the initial path search we defined the translational velocity as a function of the maximum velocity and the roughness. Since we now have a better understanding of the risk of a robot state, we use the safety cost c_{safety} of the previous section to adjust the velocity.

$$t_v(i, j) = \xi_v \cdot \frac{d_{ij}}{\max(v_{\min}, (1 - w \cdot c_{\text{safety}}(i, j)) \cdot v_{\max})}, \quad (10.8)$$

where d_{ij} is the distance between x_i and x_j , and v_{\min} and v_{\max} are the minimal and maximal forward velocity, respectively. $w \in [0, 1]$ is the safety weight of the second planning phase and controls the impact of the state safety on the planning. This term is also normalized to $[0, 1]$ using $\xi_v = \frac{1}{\max_{i,j} t_v}$.

The maximal physically possible rotational velocity depends on the ground contact of the robot's actuators. For instance, if the actuators of the Telex robot (Fig. 10.1) are completely stretched with a total length of 1.2 m, rotating is practically impossible. We performed several experiments to determine the rotational velocities given the actuator positions and composed a look-up table (for continuous values this would be a function). Please note, the quantities are sufficient for planning at this level of detail, even though they are experimentally established on flat ground and different surface frictions are neglected. The essential information these values provide is that the maximum rotational velocity varies with the robot configuration, and some configurations are more qualified for turning than others. Using this

information the time required for turning from state x_i to state x_j is given by

$$t_\omega = t_\omega(i, j) = \xi_\omega \cdot \frac{|\theta_i - \theta_j|}{\frac{1}{2}(\omega(a_i) + \omega(a_j))}, \quad (10.9)$$

where θ_i and θ_j are the orientations of the poses corresponding to x_i and x_j , respectively. $\omega(\cdot)$ provides the rotational velocity of an actuator configuration, i.e. in our case the value of the look-up table. Further, a_i and a_j are the actuator configurations of x_i and x_j , respectively. $\xi_\omega = \frac{1}{\max_{i,j} t_\omega}$ normalizes the time values to $[0, 1]$.

Similar to the previous formulas, we define the cost of the actuator movements as the normalized movement time.

$$t_a = t_a(i, j) = \xi_a \cdot \max_k \left(\frac{|a_{i,k} - a_{j,k}|}{v_k} \right), \quad (10.10)$$

where $a_{i,k}$ and $a_{j,k}$ are the values of the k th actuator of x_i and x_j , respectively. v_k is the actuator velocity of the k th actuator. Again, $\xi_a = \frac{1}{\max_{i,j} t_a}$ scales the values to $[0, 1]$.

To save execution time we would like the system to perform actions simultaneously. Therefore, the formulation of the time value of our cost function favors simultaneous execution by implementing the triangle inequality. We omitted the state variables i and j for simplicity.

$$c_{\text{time}}(i, j) = \frac{t_v^2 + t_\omega^2 + (1 - w \cdot c_{\text{safety}})^2 \cdot t_a^2}{t_v + t_\omega + (1 - w \cdot c_{\text{safety}}) \cdot t_a}. \quad (10.11)$$

The time value measuring the actuator movements is scaled by the inverse safety cost since in hazardous areas and especially in unstable states we want the system to apply actuator adjustments which improve the robot state even if they are time consuming. Note, this cost function is only applied in rough regions and may not be applicable in flat areas. In flat areas actuator movements are generally unnecessary and solely introduce costs, thus, should not be favored.

The cost function used to determine the edge weights consists of the safety metric and the time measure. The trade-off between safety and speed can be adjusted by $w \in [0, 1]$ for different mission goals. Given two neighboring states $x_i, x_j \in X_s$ the cost of the transition is defined by

$$c(i, j) = w \cdot c_{\text{safety}}(i, j) + (1 - w) \cdot c_{\text{time}}(i, j). \quad (10.12)$$

Note, since all quantities are normalized, the value of the cost function is also bound to $[0, 1]$. It is important to state here, that even though we introduced several normalization terms, the only arbitrary values are the tube size, the minimal velocity v_{\min} and the safety weights w_{is} and w . All other values are identified by the robot model and its abilities. Using a graph-based model with the defined edge weights

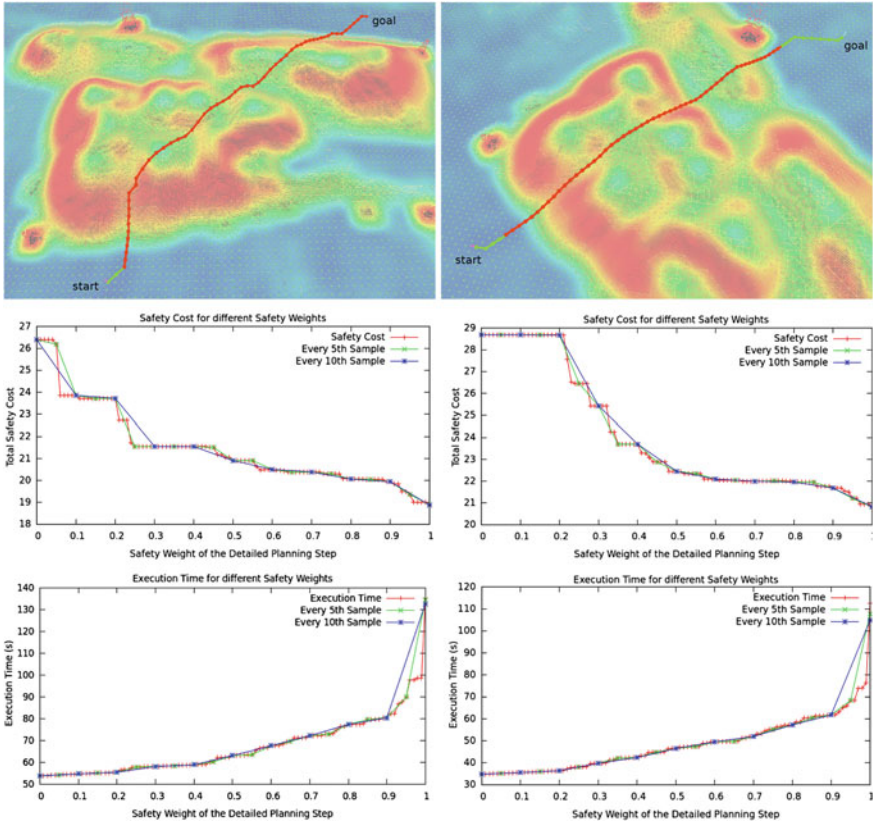


Fig. 10.7 Influence of the safety weight of the detailed motion planning step on the execution time and safety of the path. The *top* images show the two paths used for evaluation, the *second row* shows the safety costs and the *third row* the execution time with respect to the safety weight

allows us to perform a Dijkstra search on the state graph G_S to find the most stable, yet, fastest path to the goal considering the complete robot state.

While the safety weight of the initial path search determines the primary course of the final path, the safety weight of the detailed motion planning step w influences the driving velocity and the applied actuator configurations. With growing importance of the robot safety the translational velocity decreases and changing actuator positions becomes cheaper. Also, the time required to move the actuators amplifies the execution time. Therefore, with an increasing safety weight the execution time rises and the safety cost declines (Fig. 10.7). For these experiments we restricted the detailed motion planning to the initial path in order to prevent the path adjustments within the tube to disturb the results and to build a common basis for the comparison.

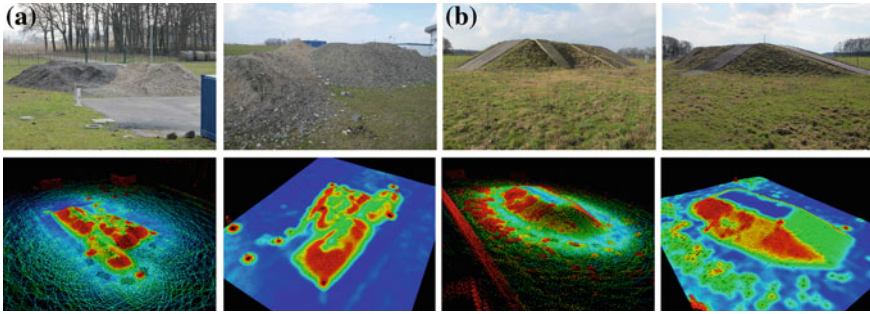


Fig. 10.8 The two real-world scenarios. The images show two pictures of a hill of rubble and the “testing hill”, the point clouds and the processed maps. The sizes of the maps are $36.4 \text{ m} \times 30.45 \text{ m}$ and $43.95 \text{ m} \times 32.95 \text{ m}$, respectively. **a** First Scenario: Hill of Rubble. **b** Second Scenario: “Testing Hill”

10.7 Experimental Results

In this section we present tests on real-world maps with a real robot to prove that our metrics and plans are valid and executable. To the best of the authors’ knowledge, there are no other approaches which aim at overcoming arbitrary obstacles with similar reconfigurable robots. Hence, we do not provide a comparison with other approaches. However, it would be possible to compare single aspects of our metrics but we consider this unfit to provide any insight of how well the metrics are suited as a whole for obstacle traversal compared to others.

We performed tests with the Telerob Telemax robot (Fig. 10.1) in two different environments (Fig. 10.8) to prove that the plans proposed by our motion planner are valid and executable by a real robot. Both maps were recorded using a laser range finder and were subsequently filled and smoothed to facilitate planning. Their sizes are $36.4 \text{ m} \times 30.45 \text{ m}$ and $43.95 \text{ m} \times 32.95 \text{ m}$ respectively. For these tests we used the following values: The robot’s maximal traversable height is $h_{max} = 0.5 \text{ m}$. In high risk areas it is allowed to drive $v_{min} = 0.2 \text{ m/s}$ and on flat ground $v_{max} = 1.2 \text{ m/s}$. The discretization of the maps was 5 cm and the window size of the filters is 20×20 cells. The resolution of the motion graph was 30 cm (half the robot length), and we considered 8 orientations in each point (corresponding to a resolution of 45°). By taking half the robot length we do not need to perform intermediate validity tests between two neighboring positions. With respect to the actuator values, both front and both rear actuators were required to be the same. Further, the positions were limited to $[-45^\circ, 45^\circ]$ in steps of 15° . The safety weights were set to $w_{is} = 0.75$ and $w = 0.5$.

On the two outdoor environments, we performed several planning queries of which we present two, one for each map. Figure 10.9 shows the planned paths and example pictures of the execution by our robot. In the first scenario (Fig. 10.9a) the robot had to cross the hill of rubble through the dips, avoiding the high risk elevations. The



Fig. 10.9 Results of the real-world experiments. The first images depict the plan given to the robot. The other images show the robot during the execution. If the robot safety permits, configurations are sometimes applied in advanced or while in motion to avoid stop-and-go movement. **a** First Scenario: Hill of Rubble. **b** Second Scenario: “Testing Hill”

robot was able to follow the proposed path while the planned actuator configurations prevented the robot from falling over. Problems during the execution were related to the small-grained material of the rubble, which caused the robot to slip casually. Equally, our robot traversed the “testing hill” of the second scenario given the motion plan (Fig. 10.9b). The localization was solely based on GPS. This, in combination with the inaccuracy of the map, made it difficult for the controller to determine which part of the plan must be executed.

However, in general, the robot was able to execute all plans of our motion planning algorithm and successfully traversed the obstacles. Further, the proposed configurations proved to be suited to ensure the safety of the robot. Problems during the execution were related to terrain parameters or to inaccuracy of the sensor data.

10.8 Conclusions and Future Work

In this chapter we present different metrics used by our hierarchical motion planner for reconfigurable robots on rough terrain. By introducing a two-phase planner we are able to use two different cost functions: one to find a low-risk path to the goal and another suited to achieve safe robot configurations along the path. The metrics include a roughness quantification, the robot’s stability and traction and the time required for position and configuration changes. Several experiments prove the validity and applicability of our measures.

We will concentrate future work on a more comprehensive search of the state space by, for instance, using optimal sampling-based algorithms for the detailed motion planning. We also focus on overcoming more complex obstacles with sharp edges of heights up to 40 cm. This will require a more detailed identification of the robot's contact with the surface to achieve a more accurate estimation of the robot pose. Also new metrics and a more sophisticated controller may be needed to cope with the arising challenges.

References

1. Jacoff, A.S., Downs, A.J., Virts, A.M., Messina, E.R.: Stepfield pallets: repeatable terrain for evaluating robot mobility. In: Performance Metrics for Intelligent Systems (PerMIS) Workshop (2008)
2. Molino, V., Madhavan, R., Messina, E., Downs, A., Balakirsky, S., Jacoff, A.: Traversability metrics for rough terrain applied to repeatable test methods. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (2007)
3. Iagnemma, K., Dubowsky, S.: Rough terrain motion planning. In: Mobile Robots in Rough Terrain—Estimation, Motion Planning, and Control with Application to Planetary Rovers, pp. 51–79. Springer Tracts in Advanced Robotics (2004)
4. Howard, T.M., Kelly, A.: Optimal rough terrain trajectory generation for wheeled mobile robots. *Int. J. Robot. Res.* **26**, 141–166 (2007)
5. Iagnemma, K., Kang, S., Shibly, H., Dubowsky, S.: Online terrain parameter estimation for wheeled mobile robots with application to planetary rovers. *IEEE Trans. Rob.* **20**, 921–927 (2004)
6. Seraji, H.: Traversability index: a new concept for planetary rovers. In: IEEE International Conference on Robotics and Automation (ICRA) (1999)
7. Howard, A., Seraji, H., Tunstel, E.: A rule-based fuzzy traversability index for mobile robot navigation. In: IEEE International Conference on Robotics and Automation (ICRA) (2001)
8. Garcia, E., Estremera, J., de Santos, P.G.: A comparative study of stability margins for walking machines. *Robotica* **20**, 595–606 (2002)
9. Miro, J., Dumonteil, G., Beck, C., Dissanayake, G.: A kyno-dynamic metric to plan stable paths over uneven terrain. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (2010)
10. Hirose, S., Tsukagoshi, H., Yoneda, K.: Normalized energy stability margin and its contour of walking vehicles on rough terrain. In: IEEE International Conference on Robotics and Automation (ICRA) (2001)
11. Magid, E., Ozawa, K., Tsubouchi, T., Koyanagi, E., Yoshida, T.: Rescue robot navigation: static stability estimation in random step environment. In: Carpin, S., Noda, I., Pagello, E., Reggiani, M., von Stryk, O. (eds.) *Simulation, Modeling and Programming for Autonomous Robots*. Lecture Notes in Computer Science, vol. 5325, pp. 305–316, Springer, Heidelberg (2008)
12. Dornhege, C., Kleiner, A.: Behavior maps for online planning of obstacle negotiation and climbing on rough terrain. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (2007)
13. Rusu, R.B., Sundaresan, A., Morisset, B., Hauser, K., Agrawal, M., Latombe, J.C., Beetz, M.: Leaving flatland: efficient real-time three-dimensional perception and motion planning. *J. Field Rob.* **26**, 841–862 (2009)
14. Brunner, M., Bruggemann, B., Schulz, D.: Towards autonomously traversing complex obstacles with mobile robots with adjustable chassis. In: International Carpathian Control Conference (2012)

15. Brunner, M., Brueggemann, B., Schulz, D.: Autonomously traversing obstacles: metrics for path planning of reconfigurable robots on rough terrain. In: International Conference on Informatics in Control, Automation and Robotics (ICINCO) (Best Paper Award) (2012)
16. Messuri, D.A.: Optimization of the locomotion of a legged vehicle with respect to maneuverability. PhD thesis, Ohio State University (1985)

Chapter 11

A Combined Direct and Indirect Adaptive Control Scheme for a Wheeled Mobile Robot using Multiple Models

Altan Onat and Metin Ozkan

Abstract This chapter presents a method about trajectory tracking control of a nonholonomic wheeled mobile robot. The main focus of the chapter is to improve the transient response for the trajectory tracking control of mobile robots including dynamic parameter uncertainties. An adaptive combined direct and indirect control scheme is used for compensation of tracking errors in case of dynamic parameter uncertainties. The transient behavior for the adaptive tracking control is improved by a multiple models approach. The overall control system includes both a kinematic and dynamic controller. The kinematic controller produces linear and angular velocities required for mobile robot to track desired trajectory. The combined direct and indirect adaptive dynamic controller with adaptive multiple identification models takes these velocities as inputs and produces torques that will be applied to the robot. Simulation results indicate effectiveness of the proposed control scheme.

Keywords Combined direct and indirect adaptive control · Trajectory tracking control · Mobile robots · Multiple models approach

11.1 Introduction

Tracking control of wheeled mobile robots (WMR) is one of the most attractive research areas for the several decades, and many wheeled mobile robot models and control schemes have been presented. Generally, the aim of such schemes is either to utilize a kinematic trajectory tracking controller or to construct and integrate

A. Onat (✉)

Electrical and Electronics Engineering Department, Anadolu University, Eskisehir, Turkey
e-mail: altanonat@anadolu.edu.tr

M. Ozkan

Computer Engineering Department, Eskisehir Osmangazi University, Eskisehir, Turkey
e-mail: meozkan@ogu.edu.tr

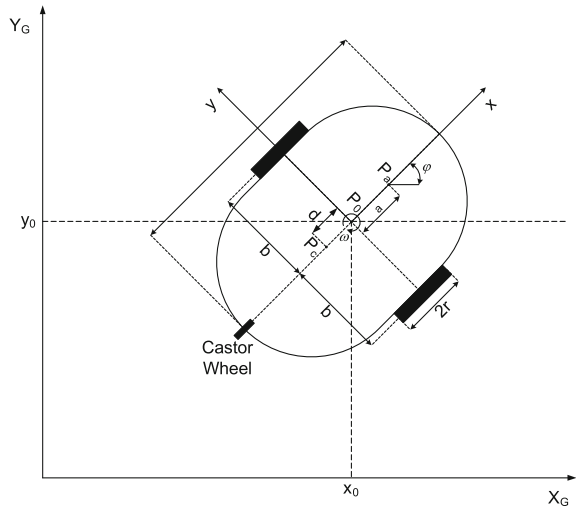
kinematic and dynamic controllers for the robot to track a desired trajectory. Kanayama et al. [1] proposed a control rule to find reasonable target linear and rotational velocities for a stable tracking control. An integrated kinematic controller and a torque controller with a dynamic extension for a nonholonomic mobile robot have been presented in [2]. Yun and Yamamoto [3] have studied feedback linearization of a wheeled mobile robot and its dynamic system. A complete dynamic model of a wheeled mobile robot which makes it suitable to consider rotational and translational velocities as control signals has been given by [4].

For tracking control of wheeled mobile robots, there are some studies which use adaptive control framework. Martins et al. [5] have proposed an adaptive controller to guide a wheeled mobile robot during trajectory tracking. In this study reference velocities are generated using a kinematic model, and then these values are processed to compensate for the robot dynamics. An adaptive trajectory tracking controller for a nonholonomic wheeled mobile robot with a nonlinear control law based on input-output feedback linearization has been proposed by [6]. Zhengcai et al. [7] has studied an adaptive kinematic controller to generate the command of velocity based on backstepping method. Similarly, a new kinematic adaptive controller integrated with a torque controller for the dynamic model of a nonholonomic wheeled mobile robot has been proposed by [8]. Pourboghraat and Karlsson [9] has used adaptive control rules for the dynamic control of nonholonomic wheeled mobile robots with unknown dynamic parameters and a fixed posture backstepping technique for tracking a reference trajectory and stabilization. Petrov [10] has proposed an adaptive dynamic based path control for a differential drive mobile robot.

In all of the studies previously mentioned provide convergence proofs. However, they did not focus on transient behavior and when the parameter errors are very large, the transient response of the system may include unacceptably large peaks. Although the system is asymptotically stable, the adaptive control approach may be inapplicable for some systems due to the transient peaks. To overcome this difficulty, enhancement of the transient response using multiple models and switching has been proposed for linear systems by [11]. Multiple models and switching for nonlinear systems have also been presented in several studies. Narendra and George [12] has presented a multiple model, switching and tuning methodology which improves the transient performance for a class of nonlinear systems. A novel approach which makes use of multiple identification models and switching based on direct adaptive control scheme has been proposed by [13]. Besides composite approach where both prediction and tracking errors are used in a combined direct and indirect adaptive control framework has been studied by [14, 15]. Ye [16] has studied a multiple model adaptive controller for nonlinear systems in parametric-strict-feedback form. Transient performance improvement for adaptive control of a class of single-input single-output (SISO) nonlinear systems, by using multiple models and switching has been considered by [17–19]. Ciliz and Narendra [20], Ciliz and Tuncay [21] have used multiple models approach in adaptive control of robotic manipulators.

The purpose of this chapter is to present an integrated kinematic and dynamic controller for trajectory tracking of a wheeled mobile robot which includes parametric uncertainties in the dynamics. A composite approach where both prediction and

Fig. 11.1 Wheeled mobile robot



tracking errors are used in a combined direct and indirect adaptive control framework with multiple identification models and switching are used for the estimation of uncertainties in the dynamics. In the literature, there are few works which makes use of the multiple models based approach for the control of wheeled mobile robots. De La Cruz and Carelli [22] have proposed a switching control for novel tracking and positioning adaptive control of wheeled mobile robots that uses multiple parameter updating laws with different gains. A method that utilizes multiple models of the robot for its identification in an adaptive and learning control framework has been presented by [23].

11.2 Kinematics and Dynamics

The model of a wheeled mobile robot (Fig. 11.1) which is subjected to m constraints may be derived as

$$M(q)\ddot{q} + C(q, \dot{q})\dot{q} = B(q)\tau + A^T(q)\lambda \tag{11.1}$$

where $q \in R^n$ is generalized coordinates, $\tau \in R^r$ is the input torque vector, $\lambda \in R^m$ is the vector of constraint forces, $M(q) \in R^{n \times n}$ is a symmetric positive-definite inertia matrix, $C(q, \dot{q}) \in R^{n \times n}$ is coriolis matrix, $B(q) \in R^{n \times r}$ is the input transformation matrix, and $A(q) \in R^{m \times n}$ is the matrix associated with the constraints (Table 11.1).

Table 11.1 Model parameters of nonholonomic wheeled mobile robot

Parameter	Description
r	Driving wheel radius
$2b$	Distance between two wheels
d	Distance point P_c from point P_0
a	Distance from P_0 to P_a
m_c	The mass of the platform without the driving wheels and the rotors of the DC motors
m_w	The mass of each driving wheel plus the rotor of its motor
I_c	The moment of inertia of the platform without the driving wheels and the rotors of the motors about a vertical axis through P_c
I_w	The moment of inertia of each wheel and the motor rotor about the wheel axis
I_m	The moment of inertia of each wheel and the motor rotor about a wheel diameter

Assuming that the velocity of P_0 is in the direction of x-axis of the local frame and there is no side slip, and considering $q = [x_0 \ y_0 \ \varphi]^T$, the following constraint with respect to P_0 is obtained

$$\dot{x}_0 \sin \varphi - \dot{y}_0 \cos \varphi = 0 \quad (11.2)$$

By writing this constraint in matrix form, matrices $A(q)$ and $S(q)$ are given by

$$A(q) = [\sin \varphi \ -\cos \varphi \ 0], \quad S(q) = \begin{bmatrix} \cos \varphi & 0 \\ \sin \varphi & 0 \\ 0 & 1 \end{bmatrix} \quad (11.3)$$

Therefore it can be written as

$$A(q) \cdot S(q) = 0 \quad (11.4)$$

It is possible to write the kinematic equation of the wheeled mobile robot in terms of the pseudo velocities vector $V(t) \in R^{n-m}$ as

$$\dot{q} = S(q) \cdot v(t) \quad (11.5)$$

where $v(t) \in [v(t) \ \omega(t)]^T$ is linear and angular velocities. The time derivative of (11.5)

$$\ddot{q} = \dot{S}(q) \cdot v + S(q) \cdot \dot{v} \quad (11.6)$$

Next, by replacing (11.5) and (11.6) in (11.1) and multiplying the result by S^T and considering (11.4), the following equation can be obtained

$$\overline{M}\dot{v}(t) + \overline{C}(v)v(t) = \overline{B}(q)\tau \quad (11.7)$$

where $\overline{M} = S^T M S$, $\overline{C} = S^T (M \dot{S} + \dot{S} S)$ and $\overline{B} = S^T B$. By denoting $\overline{B}(q)\tau$ as $\overline{\tau}$

$$\overline{M}\dot{v}(t) + \overline{C}(v)v(t) = \overline{\tau} \quad (11.8)$$

\overline{M} and \overline{C} matrices obtained as follows:

$$\overline{M} = \begin{bmatrix} m & 0 \\ 0 & I \end{bmatrix}, \overline{C} = \begin{bmatrix} 0 & m_c d \dot{\varphi} \\ -m_c d \dot{\varphi} & 0 \end{bmatrix} \quad (11.9)$$

where $m = m_c + 2m_W$ and $I = I_C + 2I_m + m_c d^2 + 2m_W b^2$. There is a parametric vector θ on dynamics that satisfies

$$\overline{M}\dot{v}(t) + \overline{C}(v)v(t) = Y(q, \dot{q}, v, \dot{v})\theta \quad (11.10)$$

where the parameters $\theta_i, i = 1, \dots, 4$ are bounded and defined as follows

$$\theta_1 = m, \theta_2 = I, \theta_3 = m_c d \quad (11.11)$$

11.3 Control Scheme

11.3.1 Kinematic Controller

In this study, a kinematic controller presented in [5] is used. The kinematic controller is based on kinematic model of the robot. Robot's kinematic model is given by

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\varphi} \end{bmatrix} = \begin{bmatrix} \cos \varphi & -a \sin \varphi \\ \sin \varphi & a \cos \varphi \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \quad (11.12)$$

where x, y are the coordinates of the point of interest P_a , and the outputs. Hence

$$h = \begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} \cos \varphi & -a \sin \varphi \\ \sin \varphi & a \cos \varphi \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} = T \begin{bmatrix} v \\ \omega \end{bmatrix} \quad (11.13)$$

$$T = \begin{bmatrix} \cos \varphi & -a \sin \varphi \\ \sin \varphi & a \cos \varphi \end{bmatrix} \quad (11.14)$$

The inverse of the matrix T is

$$T^{-1} = \begin{bmatrix} \cos \varphi & \sin \varphi \\ -\frac{1}{a} \sin \varphi & \frac{1}{a} \cos \varphi \end{bmatrix} \quad (11.15)$$

Therefore, the inverse kinematics is given by

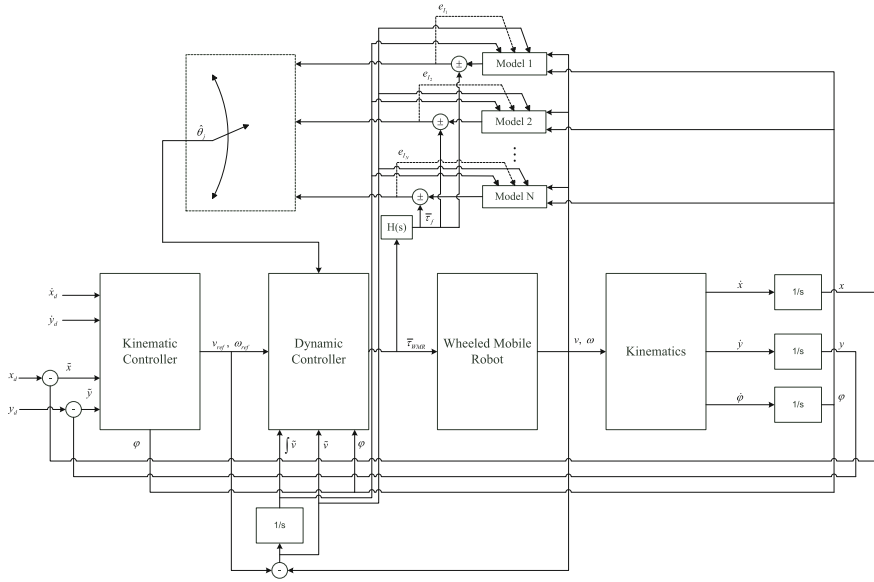


Fig. 11.2 Block diagram of the control architecture

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} \cos \varphi & \sin \varphi \\ -\frac{1}{a} \sin \varphi & \frac{1}{a} \cos \varphi \end{bmatrix} \begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} \tag{11.16}$$

and the proposed kinematic controller is given by

$$\begin{bmatrix} v_{ref} \\ \omega_{ref} \end{bmatrix} = \begin{bmatrix} \cos \varphi & \sin \varphi \\ -\frac{1}{a} \sin \varphi & \frac{1}{a} \cos \varphi \end{bmatrix} \begin{bmatrix} \dot{x}_d + I_x \tanh \left(\frac{k_x}{I_x} \tilde{x} \right) \\ \dot{y}_d + I_y \tanh \left(\frac{k_y}{I_y} \tilde{y} \right) \end{bmatrix} \tag{11.17}$$

Here, $\tilde{x} = x_d - x$, and $\tilde{y} = y_d - y$ are the current position errors in the direction of $x - axis$ and $y - axis$, respectively. $k_x > 0$ and $k_y > 0$ are the gains of the controller, $I_x \in \mathbb{R}$, and $I_y \in \mathbb{R}$ are the saturation constants, and (x, y) and (x_d, y_d) are the current and the desired coordinates of the point of interest, respectively. The purpose of this controller is to generate the reference linear and angular velocities for the dynamic controller as shown in Fig. 11.2.

11.3.2 Adaptive Dynamic Controller

Let us define a Proportional-Integral (PI) filtered velocity tracking error signal [24] as

$$s = e_v + \lambda \int e_v dt \tag{11.18}$$

where λ is a positive definite control gain and velocity tracking error is defined as

$$e_v = v_d - v \quad (11.19)$$

Taking the derivative of (11.18),

$$\dot{s} = \dot{e}_v + \lambda e_v \quad (11.20)$$

can be obtained. Considering (11.8) and adding these PI filtered error terms yields

$$\overline{M}\dot{s} + \overline{C}(v)s = Y_2(v_d, \dot{v}_d)\theta - \overline{\tau} \quad (11.21)$$

where

$$Y_2(v_d, \dot{v}_d)\theta = \overline{M}(\dot{v}_d + \lambda e_v) + \overline{C}(v)(v_d + \lambda \int e_v dt) \quad (11.22)$$

To determine the control law, and adaptive parameter update rule, consider the following Lyapunov like function

$$V = \frac{1}{2}s^T \overline{M}s + \tilde{\theta}^T \Gamma^{-1} \tilde{\theta} \quad (11.23)$$

and differentiating this function with respect to time

$$\dot{V} = \frac{1}{2}s^T \dot{\overline{M}}s + s^T \overline{M}\dot{s} + \tilde{\theta}^T \Gamma^{-1} \dot{\tilde{\theta}} \quad (11.24)$$

By taking $\overline{M}\dot{s}$ from (11.21) and substituting into (11.24), the equation becomes

$$\dot{V} = s^T (Y_2(v_d, \dot{v}_d)\theta - \overline{\tau} - \overline{C}(v)s) + \frac{1}{2}s^T \dot{\overline{M}}s + \tilde{\theta}^T \Gamma^{-1} \dot{\tilde{\theta}} \quad (11.25)$$

By choosing the control law

$$\overline{\tau} = Y_2(v_d, \dot{v}_d)\hat{\theta} + K_v s \quad (11.26)$$

and inserting this control law to (11.25) the following equation can be obtained

$$\dot{V} = s^T Y_2(v_d, \dot{v}_d)\tilde{\theta} - s^T K_v s + \frac{1}{2}s^T \left(\dot{\overline{M}} - 2\overline{C}(v) \right) s + \tilde{\theta}^T \Gamma^{-1} \dot{\tilde{\theta}} \quad (11.27)$$

Reader should note that the matrix $\dot{\overline{M}} - 2\overline{C}(v)$ is a skew-symmetric matrix. By choosing the following parameter update rule as

$$\dot{\tilde{\theta}} = -\Gamma \left(Y_1^T(q, \int v, v)\tilde{\tau}_f + Y_2^T(v_d, \dot{v}_d)s \right) \quad (11.28)$$

with an identification error model

$$\tilde{\tau}_f = Y_1(q, \int v, v)\tilde{\theta} \quad (11.29)$$

and inserting (11.28) into (11.27)

$$\begin{aligned} \dot{V} = & s^T Y_2(v_d, \dot{v}_d)\tilde{\theta} - s^T K_v s + \frac{1}{2}s^T \left(\dot{\bar{M}} - 2\bar{C}(v) \right) s \\ & + \tilde{\theta}^T \Gamma^{-1} \left(-\Gamma \left(Y_1^T(q, \int v, v)\tilde{\tau}_f + Y_2^T(v_d, \dot{v}_d)s \right) \right) \end{aligned} \quad (11.30)$$

where $Y_1(q, \int v, v)$ is the filtered regressor matrix and $\tilde{\tau}_f$ is the filtered torque term as given in [20]. Rearranging (11.29)

$$\dot{V} = -s^T K_v s + \frac{1}{2}s^T \left(\dot{\bar{M}} - 2\bar{C}(v) \right) s - \tilde{\theta}^T Y_1^T(q, \int v, v)\tilde{\tau} \quad (11.31)$$

can be obtained. By considering (11.29) and inserting it to (11.31)

$$\dot{V} = -s^T K_v s + \frac{1}{2}s^T \left(\dot{\bar{M}} - 2\bar{C}(v) \right) s - \tilde{\theta}^T Y_1^T(q, \int v, v)Y_1(q, \int v, v)\tilde{\theta} \quad (11.32)$$

can be obtained. It should be noted that \dot{V} is negative definite. It can be stated that v in (11.23) is upper bounded and that $\bar{M}(q)$ is a positive definite matrix also it can be stated that s and $\tilde{\theta}$ are bounded. Standard linear control arguments can be used to state that e_v and $\int e_v$ are bounded. Since $e_v, \int e_v, s, \tilde{\theta}$ are bounded it can be shown that \dot{s} and \dot{V} are also bounded. The reader should note that since $\bar{M}(q)$ is lower bounded, it can be stated that V is also lower bounded. Since \dot{V} is lower bounded, V is negative definite and \dot{V} is bounded, the Barbalat's Lemma can be used to state that

$$\lim_{t \rightarrow \infty} \dot{V} = 0 \quad (11.33)$$

which means that by Rayleigh-Ritz Theorem

$$\lim_{t \rightarrow \infty} \lambda_{\min} \{K_v\} \|s\|^2 = 0 \text{ or } \lim_{t \rightarrow \infty} s = 0 \quad (11.34)$$

Using the standard linear control arguments, the following can be written

$$\lim_{t \rightarrow \infty} \int e_v = 0 \text{ and } \lim_{t \rightarrow \infty} e_v = 0 \quad (11.35)$$

11.3.3 Adaptive Dynamic Controller with Multiple Models

At any given instant identification errors of all the models are available, but only one of the torque vectors will be chosen as inputs. Identification models have the following structure

$$\hat{\tau}_j = \hat{M}_j \dot{v}(t) + \hat{C}_j(v)v(t) = Y(q, \dot{q}, v, \dot{v})\hat{\theta}_j \quad (11.36)$$

where $j = 1, \dots, N$, $\hat{\theta}_j$ denoting the parameter estimate vector and $Y(q, \dot{q}, v, \dot{v})$ is the non-linear regressor matrix. The regressor matrix common to all models, but the parameter vector $\hat{\theta}_j$ has different initializations chosen from a given compact parameter set. Using the filtering technique previously mentioned nonlinear regressor matrix without acceleration signal can be obtained and will be denoted as $Y_1(q, \int v, v)$. Each model is updated using simple gradient algorithm as it is in single model case:

$$\dot{\hat{\theta}}_j = -\Gamma(Y_1^T(q, \int v, v)\tilde{\tau}_{fj} + Y_2^T(v_d, \dot{v}_d)s) \quad (11.37)$$

based on the error model which is defined as,

$$\tilde{\tau}_{fj} = e_{fj} = \bar{\tau}_f - \hat{\tau}_{fj} = Y_1(q, \int v, v)\tilde{\theta}_j \quad (11.38)$$

where $\tilde{\tau}_f$ is the filtered torque prediction error. $Y_2(v_r, \dot{v}_r)$ is the regressor matrix common to all models which is given in (11.22). The torque vector $\bar{\tau}_j$ of j th identification model is given as:

$$\bar{\tau}_j = Y_2(v_d, \dot{v}_d)\hat{\theta}_j + K_v s \quad (11.39)$$

Substituting (11.39) and (11.21) into (11.8), the closed loop dynamics can be obtained as:

$$\bar{M}_j \dot{s} + \bar{C}_j(v)s + K_v s = \tilde{M}_j(\dot{v}_d + \lambda e_v) + \tilde{C}_j(v)(v_d + \lambda \int e_v) \quad (11.40)$$

which can further be written as

$$\bar{M}_j \dot{s} + \bar{C}_j(v)s + K_v s = Y_2(v_d, \dot{v}_d)\tilde{\theta}_j \quad (11.41)$$

At any given instant identification errors of the N models are available, but only one of the torque vectors $\bar{\tau}_j$ will be chosen as the input to the WMR.

In order to choose a switching criterion, first a permissible switching sequence and a switching rule must be given as in [14, 20]. A finite or infinite sequence $T_i : T_i \in R^+$ is defined as a switching sequence if $T_0 = 0$ and $\forall i, T_i < T_{i+1}$. Additionally, if there is a number $T_{\min} > 0$ such that $\forall i, T_{i+1} - T_i \geq T_{\min}$, then the sequence is called permissible switching scheme.

A switching rule is a function of time that takes on values in the set $1, \dots, N$ is constant in $[T_i, T_{i+1})$ and is continuous from right. In other words, a function $h(t) : R_+ \rightarrow 1, \dots, N$ is called switching rule, if there exists a switching sequence $T_{i=0}$ such that if $t \in [T_i, T_{i+1})$ for some $i < \infty$, then $h(t) = h(T_i)$. With this definition torque input in (11.21) can be defined as:

$$\bar{\tau}(t) = \bar{\tau}_{h(t)}(t) \quad t \geq 0. \quad (11.42)$$

The torque vector combined with a permissible switching rule given as

$$\bar{\tau}_{h(t)} = Y_{2_{h(t)}}(v_d, \dot{v}_d)\hat{\theta}_j + K_v s \quad (11.43)$$

For the proof of stability, the same procedure will be followed as in the single model case. The additional requirement is that under any permissible switching rule, all signals should remain bounded. We have a Lyapunov like function

$$V_j = \frac{1}{2}s^T \bar{M}_j s + \tilde{\theta}_j^T \Gamma^{-1} \tilde{\theta}_j \quad (11.44)$$

By following the same procedure as in single model case the derivative of (11.44) can be obtained as in the following equation

$$\dot{V}_j = -s^T K_v s + \frac{1}{2}s^T \left(\dot{\bar{M}}_j - 2\bar{C}_j(v) \right) s - \tilde{\theta}_j^T Y_1^T(q, \int v, v) Y_1(q, \int v, v) \tilde{\theta}_j \quad (11.45)$$

Note that the matrix $\dot{\bar{M}}_j - 2\bar{C}_j(v)$ is a skew-symmetric matrix. As in the single model case \dot{V}_j is negative definite. It can be stated that V_j in (11.44) is upper bounded and that $\bar{M}_j(q)$ is a positive definite matrix. It can be stated that s and $\tilde{\theta}_j$ are bounded. Standard linear control arguments can be used to state that e_v and $\int e_v$ are bounded. Since $e_v, \int e_v, s, \tilde{\theta}$ are bounded it can be shown that \dot{s} and \dot{V}_j are also bounded. The reader should note that since $\bar{M}_j(q)$ is lower bounded, it can be stated that V_j is also lower bounded. Since \dot{V}_j is lower bounded, V_j is negative definite and \dot{V}_j is bounded, the Barbalat's Lemma can be used to state that

$$\lim_{t \rightarrow \infty} \dot{V}_j = 0 \quad (11.46)$$

which means that by Rayleigh-Ritz Theorem

$$\lim_{t \rightarrow \infty} \lambda_{\min} \{K_v\} \|s\|^2 = 0 \text{ or } \lim_{t \rightarrow \infty} s = 0 \quad (11.47)$$

Using the control arguments as in single model case the following can be written

$$\lim_{t \rightarrow \infty} \int e_v = 0 \text{ and } \lim_{t \rightarrow \infty} e_v = 0 \quad (11.48)$$

11.3.4 Proof of Stability for the Kinematic Controller

For the details of the proof, readers may read [5]. By considering (11.13) and (11.14):

$$\begin{bmatrix} \dot{\tilde{x}} \\ \dot{\tilde{y}} \end{bmatrix} + \begin{bmatrix} I_x & 0 \\ 0 & I_y \end{bmatrix} \begin{bmatrix} \tanh\left(\frac{k_x}{I_x}\tilde{x}\right) \\ \tanh\left(\frac{k_y}{I_y}\tilde{y}\right) \end{bmatrix} = \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \end{bmatrix} \quad (11.49)$$

One can see that the error vector ε can also be written as Te , where e is the velocity tracking error and matrix T is defined previously. Rewriting (11.49)

$$\dot{\tilde{h}} + L(\tilde{h}) = Te, \quad L(\tilde{h}) = \begin{bmatrix} I_x & 0 \\ 0 & I_y \end{bmatrix} \begin{bmatrix} \tanh\left(\frac{k_x}{I_x}\tilde{x}\right) \\ \tanh\left(\frac{k_y}{I_y}\tilde{y}\right) \end{bmatrix} \quad (11.50)$$

Now considering Lyapunov candidate function and its derivative

$$\begin{aligned} V &= \frac{1}{2}\tilde{h}^T\tilde{h}, \\ \dot{V} &= \tilde{h}^T\dot{\tilde{h}} = \tilde{h}^T(Te - L(\tilde{h})) \end{aligned} \quad (11.51)$$

and a sufficient condition for $\dot{V} < 0$ can be expressed as

$$\tilde{h}^T L(\tilde{h}) > \left| \tilde{h}^T Te \right| \quad (11.52)$$

For small values of the control error \tilde{h} following can be written

$$L(\tilde{h}) \approx K_{xy}\tilde{h}, \dots K_{xy} = \begin{bmatrix} k_x & 0 \\ 0 & k_y \end{bmatrix} \quad (11.53)$$

Now the sufficient condition for $\dot{V} < 0$ can be written as

$$\tilde{h}^T K_{xy}\tilde{h} > \left| \tilde{h}^T Te \right|, \min(k_x, k_y) \|\tilde{h}\|^2 > \|\tilde{h}\| \|Te\|, \|\tilde{h}\| > \frac{\|Te\|}{\min(k_x, k_y)} \quad (11.54)$$

It is shown that e tend to zero as $t \rightarrow \infty$, which implies that condition in (11.43) is verified for any value of \tilde{h} . Thus, $\tilde{h}(t) \rightarrow 0$ as $t \rightarrow \infty$.

11.3.5 Switching Criterion

A cost function is considered in the form

$$J_j(t) = e_{I_j}(t)^T G_1 e_{I_j}(t) + \int_0^t e^{-\lambda(t-\tau)} e_{I_j}(t)^T G_2 e_{I_j}(t) d\tau \quad (11.55)$$

where J_j is the cost function of the j th model, e_{I_j} is the identification error associated with the j th model, $G_1, G_2 \in R^{n \times n}$ are positive (semi)-definite weight matrices and $\lambda \geq 0$ is a scalar forgetting factor. J_a is denoted as the cost function of the current model. If $J_a(t) > J_j(t)$ with defined switching sequence, it means that adaptive model must be switched to the j th model according to the switching criterion.

11.4 Simulations

In simulations, a wheeled mobile robot tracks a trajectory given by

$$x_r = x_g + 2\omega_r t, y_r = y_g + \sin(2\omega_r t) \quad (11.56)$$

where $x_g = 0, y_g = 2$, and $\omega_r = 0.04$. Initially robot $x_0 = 0$ and $y_0 = 1$ and robot has zero velocities and $\varphi = \pi/2$.

Wheeled mobile robot parameters taken as $I_m = 0.0025 \text{ Kg.m}^2, I_c = 15.625 \text{ Kg.m}^2, r = 0.15 \text{ m}, b = 0.75 \text{ m}, a = 0.3 \text{ m}, d = 0.3 \text{ m}, L = 0.1 \text{ m}, m_w = 1 \text{ Kg}, m_c = 36 \text{ Kg}, I_w = 0.005 \text{ Kg.m}^2, K_v = 10, \lambda = \text{diag}(10, 10), \Gamma = \text{diag}(2, 2, 2), \alpha = 1, k_x = 10, k_y = 10, I_y = 1, I_x = 1$. Switching sequence has a time step of 5 ms.

The real values of the unknown parameters are $\theta = [38 \ 19.95 \ 10.8]^T$, and the initial estimates for the parameters are $\hat{\theta} = [20 \ 7 \ 3]^T$.

In order to show effectiveness of the developed solution, ten identification models have been chosen as

$$\begin{aligned} \hat{\theta}_1 &= [29 \ 11 \ 5]^T, \hat{\theta}_2 = [32 \ 14 \ 7]^T, \hat{\theta}_3 = [35 \ 17 \ 9]^T, \hat{\theta}_4 = [38 \ 20 \ 11]^T, \\ \hat{\theta}_5 &= [41 \ 23 \ 13]^T, \hat{\theta}_6 = [44 \ 26 \ 15]^T, \hat{\theta}_7 = [47 \ 29 \ 17]^T, \hat{\theta}_8 = [50 \ 32 \ 19]^T, \\ \hat{\theta}_9 &= [53 \ 35 \ 21]^T, \hat{\theta}_{10} = [56 \ 38 \ 23]^T. \end{aligned}$$

It can be seen from the figures that proposed control approach enhances the performance of both velocity tracking and trajectory tracking.

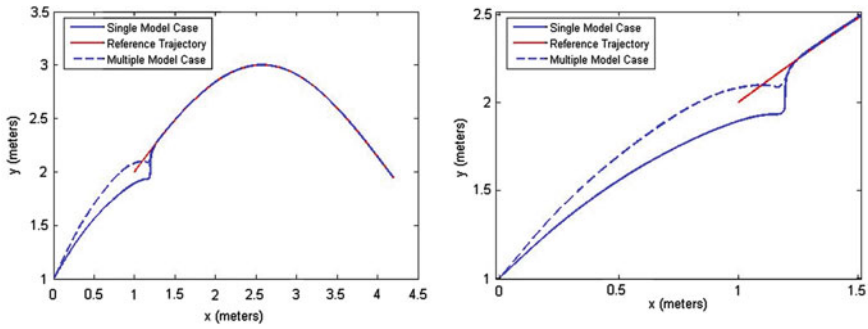


Fig. 11.3 Trajectories for single and multiple model case: (left) wide view, (right) close view

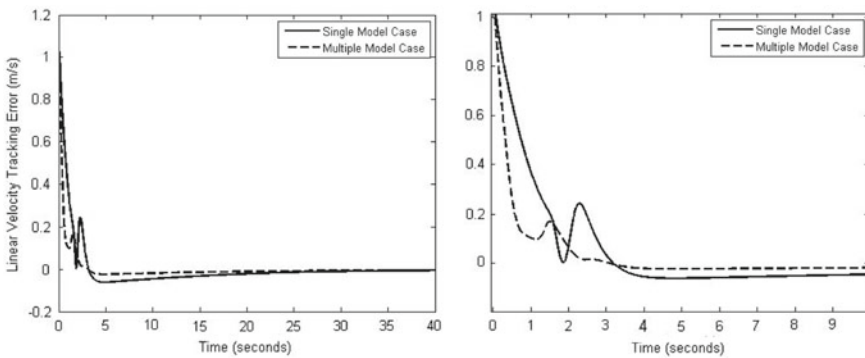


Fig. 11.4 Linear velocity tracking error for both single and multiple model case: (left) wide view, (right) close view

11.5 Conclusions

A combined direct and indirect adaptive control scheme with a multiple models approach is proposed for the trajectory tracking of a WMR. The control scheme includes two stage controllers. The dynamic controller provides fast velocity tracking under parameter uncertainties. The kinematic controller provides the velocity profile needed for the trajectory tracking of the WMR in Cartesian coordinates. The stability of the overall control system was proved. The main focus was to improve the transient response for the proposed adaptive control scheme. Simulations show that the proposed control system is applicable to the WMR and it significantly enhances the transient behavior during the trajectory tracking (Figs. 11.3, 11.4, 11.5, 11.6 and 11.7).

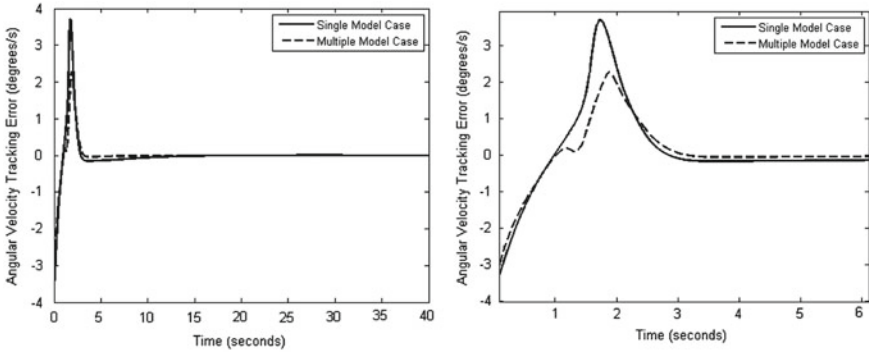


Fig. 11.5 Angular velocity tracking error for both single and multiple model case: (left) wide view, (right) close view

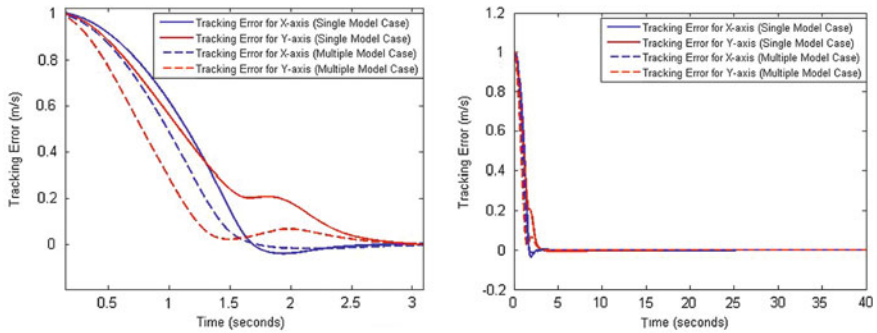
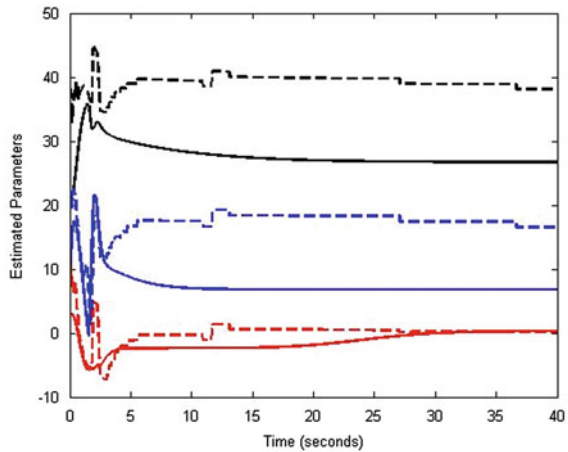


Fig. 11.6 Tracking error on the x and y axis: (left) wide view, (right) close view

Fig. 11.7 Estimated parameters: (solid) single model case, (dashed) multiple model case



References

1. Kanayama, Y., Kimura, Y., Miyazaki, F., Noguchi, T.: A stable tracking control method for an autonomous mobile robot. In: International Conference on Robotics and Automation, USA (1990)
2. Yun, X., Yamamoto, Y.: On feedback linearization of mobile robots. University of Pennsylvania, department of Computer and information science Technical reports (1992)
3. Fierro, R., Lewis, F.L.: Control of a nonholonomic mobile robot. Backstepping kinematics into dynamics. In: Proceedings of the 34th Conference on Decision and Control. New Orleans, LA (1995)
4. De La Cruz, C., Carelli, R., Bastos, T.F.: Switching adaptive control of mobile robots. In: IEEE International Symposium on Industrial Electronics (2008)
5. Martins, F.N., Celeste, W.C., Carelli, R., Sarcinelli-Filho, M., Bastos-Filho, T.: An adaptive dynamic controller for autonomous mobile robot trajectory tracking. *Control Eng. Pract.* **16**, 1354–1363 (2008)
6. Shojaei, K., Shahri, A.M., Tarakameh, A., Tabibian, B.: Adaptive trajectory tracking control of a differential drive wheeled mobile robot. *Robotica* **29**(3), 391–402 (2011)
7. Zhengcai, C., Yingtao, Z., Qidi, W.: Adaptive trajectory tracking control for a nonholonomic mobile robot. *Chin. J. Mech. Eng.* **24**(3), 1–7 (2011)
8. Fukao, T., Nakagawa, H., Adachi, N.: Adaptive tracking control of a nonholonomic mobile robot. *IEEE Trans. Robot. Autom.* **16**(5), 609–615 (2000)
9. Pourboghra, F., Karlsson, M.P.: Adaptive control Of dynamic mobile robots with nonholonomic constraints. *Comput. Electr. Eng.* **28**(4), 241–253 (2002)
10. Petrov, P.: Modeling and adaptive path control of a differential drive mobile robot. In: Proceedings of the 12th WSEAS International Conference on Automatic Control, Modelling and Simulation (2010)
11. Narendra, K.S., Balakrishnan, J.: Adaptive control using multiple models. *IEEE Trans. Autom. Control* **42**(2), 171–187 (1997)
12. Narendra, K.S., George, K.: Adaptive control of simple nonlinear systems using multiple models. In: Proceedings of the American Control Conference Anchorage (2002)
13. Cezayirli, A., Ciliz, M.K.: Transient performance enhancement of direct adaptive control of nonlinear systems using multiple models and switching. *IET Control Theory Appl.* **1**(6), 1711–1725 (2007)
14. Ciliz, K., Narendra, K.S.: Intelligent control of robotic manipulators: a multiple model based approach. In: IEEE International Conference on Intelligent Robots and Systems 95. Human robot interaction and cooperative robots, pp. 422–427 (1995)
15. Ciliz, K., Cezayirli, A.: Combined direct and indirect control of robot manipulators using multiple models. In: Proceedings of the 2004 IEEE Conference on Robotics, Automation and Mechatronics, Singapore (2004)
16. Ye, X.: Nonlinear adaptive control using multiple identification models. *Syst. Control Lett.* **3**, 488–491 (2008)
17. Ciliz, M.K., Cezayirli, A.: Increased transient performance for adaptive control of feedback linearizable systems using multiple models. *Int. J. Control* **79**(10), 1205–1215 (2006)
18. Ciliz, K., Cezayirli, A.: Adaptive tracking for nonlinear plants using multiple identification models and state-feedback. In: IEEE Industrial Electronics, IECON 2006–32nd Annual Conference (2006)
19. Cezayirli, A., Ciliz, M.K.: Indirect adaptive control of non-linear systems using multiple identification models and switching. *Int. J. Control* **81**(9), 1434–1450 (2008)
20. Ciliz, K., Narendra, K.S.: Multiple model based adaptive control of robotic manipulators. In: Proceedings of the 33rd Conference on Decision and Control, Lake Buena Vista, FL (1994)
21. Ciliz, K., Tuncay, M.Ö.: Comparative experiments with a multiple model based adaptive controller for SCARA type direct drive manipulator. *Robotica* **23**(6), 721–729 (2005)
22. De La Cruz, C., Carelli, R.: Dynamic modeling and centralized formation control of mobile robots. In: IEEE Industrial Electronics, IECON 2006 (2006)

23. D'Amico, A., Ippoliti, G., Longhi, S.: A multiple models approach for adaptation and learning in mobile robots control. *J. Intell. Rob. Syst.* **47**, 3–31 (2006)
24. Wison, D.G., Robinett, R.D.: Robust adaptive backstepping control for a nonholonomic mobile robot. In: *IEEE International Conference on Systems, Man, and Cybernetics* (2001)

Chapter 12

Real-Time Visual Servoing Based on New Global Visual Features

Laroussi Hammouda, Khaled Kaaniche, Hassen Mekki
and Mohamed Chtourou

Abstract This chapter proposes a new approach to achieve real-time visual servoing tasks. Our contribution consists in the definition of new global visual features as a random distribution of limited set of pixels luminance. The new method, based on a random process, reduces the computation time of the visual servoing scheme and removes matching and tracking process. Experimental results validate the proposed approach and show its robustness regarding to the image content.

Keywords Visual servoing · Global visual features · Mobile robot

12.1 Introduction

Computer vision is progressively playing more important role in service robotic applications. In fact, the movement of a robot equipped with a camera can be controlled from its visual perception using visual servoing technique. The aim of the visual servoing is to control a robotic system using visual features acquired by a visual sensor [1]. Indeed, the control law is designed to move a robot so that the current visual features s , acquired from the current pose r , will reach the desired features s^* acquired from the desired pose r^* , leading to a correct realization of the task.

The control principle is thus to minimize the error $e = s - s^*$ where e is a vector containing the current values of the chosen visual information, and s^* its desired values. The basic step in image-based visual servoing is to determine the adequate set of visual features to be extracted from the image and used in the control scheme in order to obtain an optimal behavior of the robot.

L. Hammouda (✉) · K. Kaaniche · H. Mekki · M. Chtourou
Control and Energies Management Laboratory, University of Sfax, Sfax, Tunisia
e-mail: Laroussi.hmd@gmail.com

K. Kaaniche · H. Mekki
National School of Engineering of Sousse, University of Sousse, Sousse, Tunisia

In the literature several works were concerned with simple objects and the features used as input of the control scheme were generally geometric: coordinates of points, edges or straight lines [2, 3].

These geometric features have always to be tracked and matched over frames. This process has proved to be a difficult step in any visual servoing scheme. Therefore, in the last decade, the researchers are focused on the use of global visual features. In fact, in [4] the visual features considered are the luminance of all image pixels and the control law is based on the minimization of the error which is the difference between the current and the desired image.

Others works are interested in the application of image moments in visual servoing, like in [5] where the authors propose a new visual servoing scheme based on a set of moment invariants. The use of these moments ensures an exponential decoupled decrease for the visual features and for the components of the camera velocity. However this approach is restricted to binary images. It gives good results except when the object is contrasted with respect to its environment.

In [6], the authors present a new criterion for visual servoing: the mutual information between the current and the desired image. The idea consists in maximizing the information shared by the two images. This approach has proved to be robust to occlusions and to very important light variations. Nevertheless, the computation time of this method is relatively high.

The work of [7] proposes the image gradient as visual feature for visual servoing tasks. This approach suffers from a small cone of convergence. Indeed, using this visual feature, the robotic system diverges in the case of large initial displacement. Another visual servoing approach which removes the necessity of features tracking and matching step has been proposed in [8]. This method models the image features as a mixture of Gaussian in the current and in the desired image. But, using this approach, an image processing step is always required to extract the visual features.

The contribution of this chapter consists in the definition of new global visual features: random distribution of limited set of pixels luminance. Our features improve the computation time of visual servoing scheme and avoid matching and tracking step. We illustrate in this work an experimental analysis of the robotic system behavior in the case of visual servoing task based on our new approach.

This chapter is organized as follows: Sect. 12.2 illustrates our new visual features and the corresponding interaction matrix. Section 12.3 recalls the optimization method used in the building of the control law. Section 12.4 presents the way to perform visual servoing for mobile robot. Finally, experimental results are presented in Sect. 12.5.

12.2 Random Distribution of Limited Set of Pixels Luminance as Visual Features

The use of the whole image luminance as global visual features for visual servoing tasks, as in [9], requires too high computation time. Indeed, the big size of the interaction matrix related to the luminance of all image pixels leads to a very slow convergence of the robotic system.

Therefore, we propose in this chapter a new visual feature which is more efficient in terms of computation time and doesn't require any matching nor tracking step.

In fact, instead of using the luminance of all image points, we work just with the luminance of a random distribution of a limited set of image points (n pixels). Thus, the visual features, at a position r of the robot, are:

$$s_i(r) = E_I^i(r) \quad (12.1)$$

with $E_I^i(r)$ is the luminance of random set of image pixels taken at frame i .

$$E_I^i(r) = (I_1^i, I_2^i, I_3^i, \dots, I_n^i) \quad (12.2)$$

where I_k^i is the luminance of the pixel K taken randomly at the frame i .

For each new frame, we get a new random set of image pixels. Thus, the desired and the current visual features will continuously change along the visual servoing scheme. In that case, the error e will be:

$$e_i = E_I^i(r) - E_{I_*}^i(r^*) \quad (12.3)$$

where $E_I^i(r)$ represent the current visual features and $E_{I_*}^i(r^*)$ the desired ones at the frame i .

Consequently, in our method, the error used in the building of the control law is variable, it changes at each frame. This change is like a kind of mutation. Convergence to global minimum is then guaranteed.




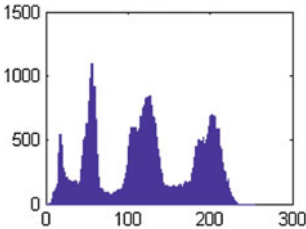
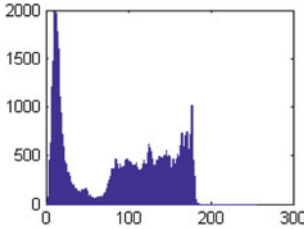
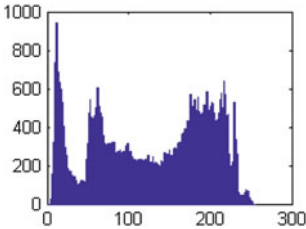
The choice of n is based on the image histogram. We take n equal to the maximum value of the current image histogram. We can then avoid the fact that the n pixels randomly chosen will have the same luminance. Hence, we guarantee the good luminance representation of the image. We note p_l the probability that the n pixels will have the same luminance. It is given by:

$$p_l = \frac{C_n^n}{C_N^n} = \frac{1}{\frac{N!}{n!(N-n)!}} \quad (12.4)$$

where n is the number of pixels deduced from the image histogram and chosen as visual features and N is the number of all image pixels. This probability is null (see Table 12.1).

Since the number n depends on the histogram of the current image, it slightly changes during the visual servoing scheme. Let us point that n is always very small

Table 12.1 Examples of images with the corresponding histograms and probabilities

Image	Histogram	n	p_l	p_s
		1089	0	0
		1979	0	0
		940	0	0

compared to the total number of image pixels (in our case 320×240). We note that the more the image is textured, the smaller n is.

Figure 12.1 shows an example of image, the luminance of all its pixels form the ancient global visual features.

The histogram of this image is illustrated on Fig. 12.2. In our approach, instead of using all image pixels, we take randomly n pixels as global visual features, with n is the maximum value of this histogram (in this example $n = 2452$ which is 3.1% of all image pixels).

After ensuring that the n pixels are good representatives of the image luminance, we can confirm that these n pixels randomly chosen will be well distributed in the image and not concentrated in one particular zone. For that, we compute the probability that the n pixels will be all in one zone z . This probability is given by:

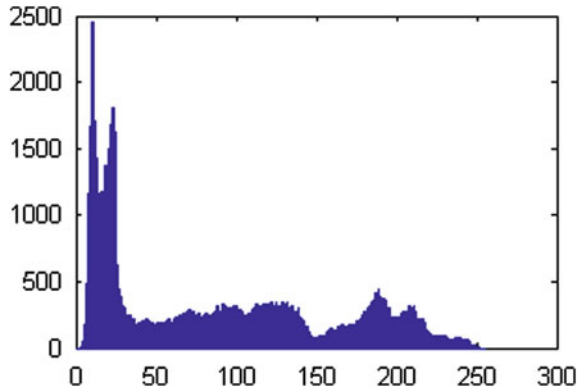
$$p_s = \frac{C_z^n}{C_N^n} = \frac{\frac{z!}{n!(z-n)!}}{\frac{N!}{n!(N-n)!}} \tag{12.5}$$

with z is the number of pixels in a compact zone of the image.

Fig. 12.1 Ancient visual features: the whole image luminance



Fig. 12.2 Image histogram (essential for the choice of n)



In our work, we take z as the half of all image pixels (Beyond this value of z we assume that good image representation is ensured).

The probability p_s is equal to zero (see Table 12.1). This proves that the n pixels chosen as visual features will always ensure good spatial representation of the scene.

We present in Table 12.1 the histograms and the probabilities (p_l and p_s) related to different images.

The visual servoing is based on the relationship between the camera motion and the consequent change on the visual features. This relationship is expressed by the well known equation [10]:

$$\dot{s} = L_s v_c \tag{12.6}$$

where L_s is the interaction matrix that links the time variation of s to the camera instantaneous velocity v_c [11].

So, after identification of the visual features, the control law requires the determination of this matrix which is at the center of the development of any visual servoing scheme. In our case, we look for the interaction matrix related to the luminance of a pixel x in the image.

The computation of this matrix is based on the optical flow constraint equation (OFCE) which is a hypothesis that assumes the temporal constancy of the luminance for a physical point between two successive images [11].

If a point x of the image realizes a displacement dx in the time interval dt , according to the previous hypothesis we have:

$$I(x + dx, t + dt) = I(x, t) \quad (12.7)$$

After development of this equation we get:

$$\nabla I^T \dot{x} + \dot{I} = 0 \quad (12.8)$$

where $\dot{I} = \frac{\partial I(x)}{\partial t}$ and ∇I is the spatial gradient of x . We know that:

$$\dot{x} = L_x v_c \quad (12.9)$$

where L_x is the interaction matrix that relates the temporal variation of x to the control law.

Using (12.8) and (12.9) we obtain:

$$\dot{I} = -\nabla I^T L_x v_c \quad (12.10)$$

So the interaction matrix that relates the temporal variation of the luminosity $I(x)$ to the control law v_c is:

$$L_{I(x)} = -\nabla I^T L_x \quad (12.11)$$

In this case, we can write the interaction matrix $L_I(x)$ in terms of the interaction matrices L_x and L_y related to the coordinates of $x = (x, y)$ and we obtain:

$$L_{I(x)} = -(\nabla I_x L_x + \nabla I_y L_y) \quad (12.12)$$

with ∇I_x et ∇I_y are the components along x and y of $\nabla I(x)$.

In the case of a mobile robotic system, we take into account just the components of L_x that correspond to three degrees of freedom: Translation along x axis, translation along z axis and rotation around y axis. Therefore, we have:

$$L_x = \begin{pmatrix} -\frac{1}{z} & \frac{x}{z} & -(1 + x^2) \end{pmatrix} \quad (12.13)$$

$$L_y = \begin{pmatrix} 0 & \frac{y}{z} & -xy \end{pmatrix} \quad (12.14)$$

where z is the depth of the point x relative to the camera frame.

We get the interaction matrix related to our new features ($L_{E_1^i}$) by combining the interaction matrices related to the n pixels randomly chosen.

Thus, the size of the interaction matrix related to our visual features ($L_{E_1^i}$) is very small compared to the size of the interaction matrix related to the whole image luminance.

12.3 The Control Law Generation

In our work we use a global photometric visual features. In this case most of classical control laws fail. Therefore, we have interest in turning the visual servoing scheme into an optimization problem to get the convergence of the mobile robot to its desired pose [12, 13]. In fact, the aim of the control law will be the minimization of a cost function which is the following:

$$\mathcal{C}(r) = (s(r) - s(r^*))^T (s(r) - s(r^*)) \quad (12.15)$$

where $s(r)$ are the current visual features ($E_1^i(r)$) and $s(r^*)$ are the desired ones ($E_{1^*}^i(r^*)$).

The cost function minimization is, essentially, based on the following step:

$$r_{i+1} = r_i \oplus d(r_i) \quad (12.16)$$

where “ \oplus ” denotes the operator that combines two consecutive frame transformations, r_i is the current pose of the mobile robot (at frame i), r_{i+1} is the next pose of the mobile robot and $d(r_i)$ is the direction of descent.

This direction of descent must ensure that $d(r_i) \nabla \mathcal{C}(r_i) < 0$. In this way, the movement of the robot leads to the decrease of the cost function.

Optimization methods depend on the direction of descent used in the building of the control law. The control law usually used in visual servoing context is given by:

$$v_c = -\lambda L_s^+ (s(r) - s(r^*)) \quad (12.17)$$

where λ is a positive scalar and L_s^+ is the pseudo inverse of the interaction matrix.

This classical control law gives good results in the case of visual servoing task based on geometric visual features [10].

Since we work with photometric visual features this classical control law fails and doesn't ensure the convergence of the robot [4]. Thus, in our work we use the control law based on the Levenberg-Marquardt approach. The control law generated to the robot, using our new features, is then given by:

$$v_c^i = -\lambda (H_{E_1^i} + \mu \text{diag}(H_{E_1^i}))^{-1} L_{E_1^i}^T e_i \quad (12.18)$$

where e_i is the error corresponding to these new features:

$$e_i = E_1^i(r) - E_{1^*}^i(r^*) \quad (12.19)$$

and with

$$H_{E_1^i} = L_{E_1^i}^T L_{E_1^i} \quad (12.20)$$

12.4 Mobile Robot Visual Control

The visual servoing approach described previously will be adapted to control a system composed from a video camera mounted on a two wheel robot.

Let's consider a two wheel robot with a pose defined by:

$$\begin{bmatrix} x \\ y \\ \theta \end{bmatrix} \quad (12.21)$$

Assuming that the motion is done without sliding, the following equations describe the kinematic behavior of the robot:

$$\begin{aligned} \dot{x} &= v_R \cos \theta \\ \dot{y} &= v_R \sin \theta \\ \dot{\theta} &= \omega_R \end{aligned} \quad (12.22)$$

where (v_R, ω_R) is the linear and angular velocity of the mobile robot.

The relation between (v_R, ω_R) and the camera instantaneous velocity is given by:

$$v_c = J_r \times \begin{pmatrix} v_R \\ \omega_R \end{pmatrix} \quad (12.23)$$

with $J_r = \begin{pmatrix} 0 & -a \\ 1 & 0 \\ 0 & -1 \end{pmatrix}$ and a is the distance between the turret on which the camera is mounted and the wheels axis.

$$\begin{pmatrix} v_R \\ \omega_R \end{pmatrix} = (J_r^T \times J_r)^{-1} \times J_r^T \times v_c \quad (12.24)$$

Having an input (v_R, ω_R) the correspondent velocities of the wheels (right and left) can be computed from:

$$\begin{pmatrix} \varphi_r \\ \varphi_l \end{pmatrix} = \begin{pmatrix} \frac{1}{r} & \frac{L}{r} \\ \frac{1}{r} & -\frac{L}{r} \end{pmatrix} \times \begin{pmatrix} v_R \\ \omega_R \end{pmatrix} \quad (12.25)$$

where L is the half distance between the robot's wheels and r is the radius of the wheel.

12.5 Experimental Results

12.5.1 Experimental Environment

We present the results of a set of experiments conducted with our visual features. All the experiments reported here have been obtained using a camera mounted on a mobile robot. In each case, the mobile robot is first moved to its desired pose r^* and the corresponding image I^* is acquired. From this desired image, we extract the desired visual features S^* . The robot is then moved to a random pose r and the initial visual features are extracted. The velocities computed, at each frame, using the control law, are sent to the robot until its convergence. The interaction matrix is calculated at each frame of the visual servoing scheme. In a first step we conduct our experiments on a virtual platform of VRML, therefore we can recuperate, at each frame, the pose of the mobile robot in terms of position along two translational axes and around one rotational axis. In a second step we validate our results on a real mobile robot (Koala robot).

12.5.2 Interpretation

During the experiments conducted on the VRML environment we take as initial positioning error: $\Delta r_{\text{int}} = (18 \text{ cm}, 12 \text{ cm}, 9^\circ)$. We illustrate the results obtained using our new visual features on Figs. 12.3 and 12.4 (first and second experiment).

Figures 12.3a and 12.4a present the initial scenes. Figures 12.3b and 12.4b depict the desired scenes. The histograms of the initial images are shown on Figs. 12.3c and 12.4c.

We choose as stopping criterion of our program the following measure: $M(r)$ which is the proportion of the number of pixels, in the error image $(I - I^*)$, whose luminance is below a certain threshold compared to the total number of image pixels.

$$M(r) = \frac{N_{\text{thres}}(r)}{N_{\text{total}}} \times 100 \quad (12.26)$$

where $N_{\text{thres}}(r)$ is the number of pixels in the error image whose luminance is below a predefined threshold at pose r of the robot and N_{total} is the total number of pixels (320×240).

In our experiments we choose the luminance value 3 as a threshold. We suppose that the convergence is achieved and the robotic system reaches its desired pose when $M(r)$ get at 98 %.

Figures 12.3d and 12.4d depict the behavior of this stopping criterion. The translational positioning errors (ΔTx , ΔTz) between the current and the desired pose during the positioning task are shown on Figs. 12.3e and 12.4e. The rotational positioning errors (ΔRy) are illustrated on Figs. 12.3f and 12.4f.

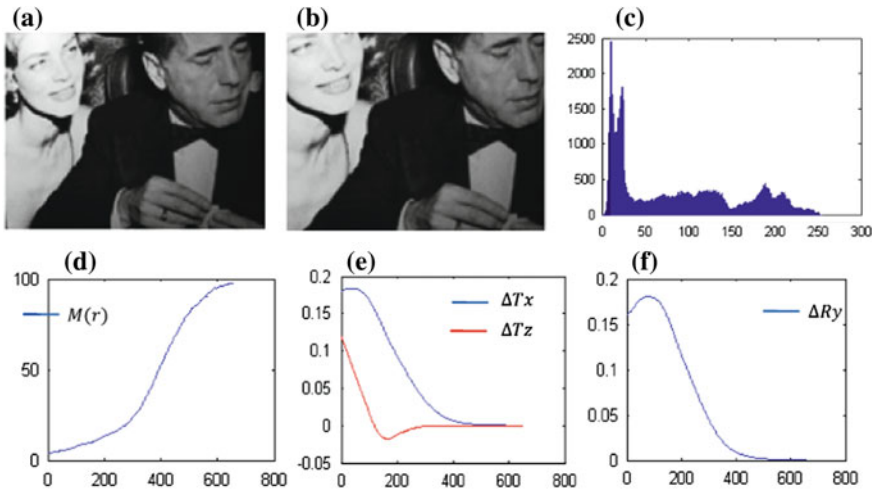


Fig. 12.3 First experiment with our new global visual features (x axis in frame number for (d), (e) and (f)): **a** Initial image, **b** Desired image, **c** Initial image histogram, **d** Stopping criterion evolution: $M(r)$ in percentage (%), **e** Translational positioning errors: ΔT_x and ΔT_z in meter (m), **f** Rotational positioning error: ΔR_y in radian (rad)

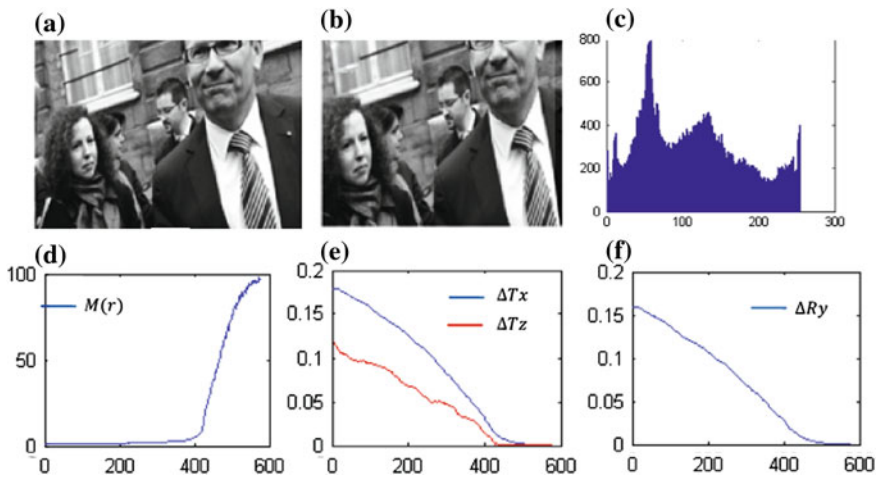
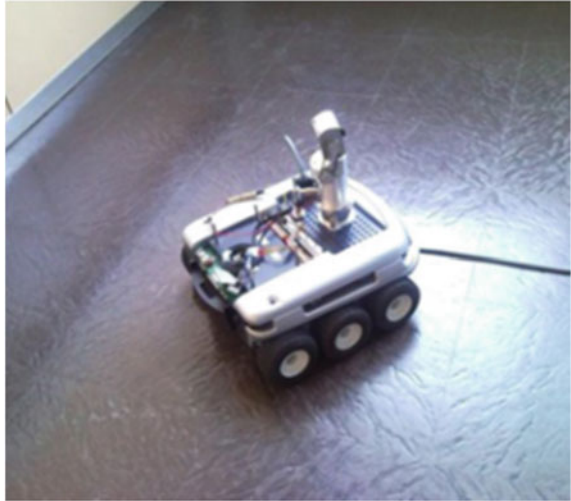


Fig. 12.4 Second experiment with our new global visual features (x axis in frame number for (d), (e) and (f)): **a** Initial image, **b** Desired image, **c** Initial image histogram, **d** Stopping criterion evolution: $M(r)$ in percentage (%), **e** Translational positioning errors: ΔT_x and ΔT_z in meter (m), **f** Rotational positioning error: ΔR_y in radian (rad)

Fig. 12.5 The Koala mobile robot



We note that the robotic system converges with good behavior using our global visual features ($s(r) = E_I^i(r)$) and it spend very less time compared to the method of [4].

Indeed, our method reduces the size of the visual features vector s . Thus, the size of the interaction matrix related to our visual features ($L_{E_I^i}$) is very small compared to the size of the interaction matrix related to the whole image luminance. Therefore, our approach is more suitable to real-time applications. As an example, the experiment of Fig. 12.3 has demonstrated that, using our approach, the computation time for each 320×240 frame does not exceed 40 ms while it is 270 ms when we work with the whole image luminance as visual features.

After using the virtual platform of VRML, we validated our new approach using the Koala mobile robot which is a differential wheeled robot (see Fig. 12.5). In fact, the visual control algorithm described was implemented, tested and validated using a real-time servoing system composed from the Koala mobile platform connected via USB with a PC that has Matlab software.

The results of the experiments conducted on the Koala are illustrated on Fig. 12.6. We remark that this mobile robot correctly converges to its desired pose using our new global visual features. The initial and the desired scene are reported respectively on Fig. 12.6a, b. The evolutions of the velocities of the two robot wheels are illustrated on Fig. 12.6c where φ_r is the right wheel and φ_l is the left one. The stopping criterion

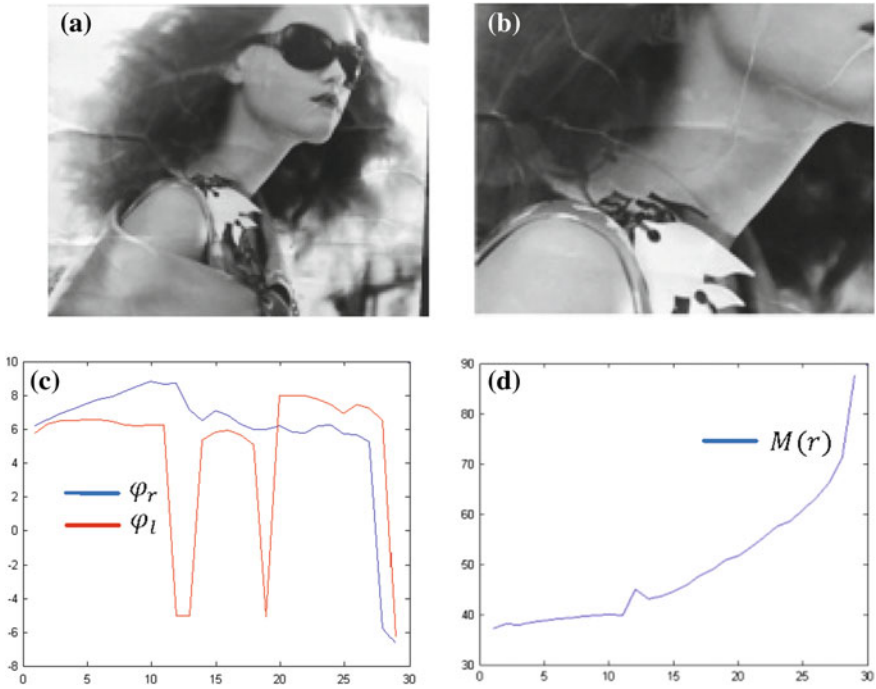


Fig. 12.6 Our global visual features (x axis in frame number). **a** Initial scene, **b** Desired scene, **c** The velocities of the two robot wheels (mm/s), **d** Stopping criterion evolution: $M(r)$ (%)

evolution is shown on Fig. 12.6d. So, we can confirm that our new visual features give good results in the case of real conditions of visual servoing task.

12.5.3 Robustness with Respect to Image Content

Our approach does not depend on the image content. In fact, the experiments demonstrate that the control law converges even in the case of low textured scenes. Figure 12.7 shows that using different types of scenes the control law converges in all the cases (we keep the same initial positioning errors). The images presented here are those used in [14].

The first column in Fig. 12.7 shows the different scenes. The second and the third column illustrate, respectively, the translational and the rotational positioning errors during the visual servoing scheme.

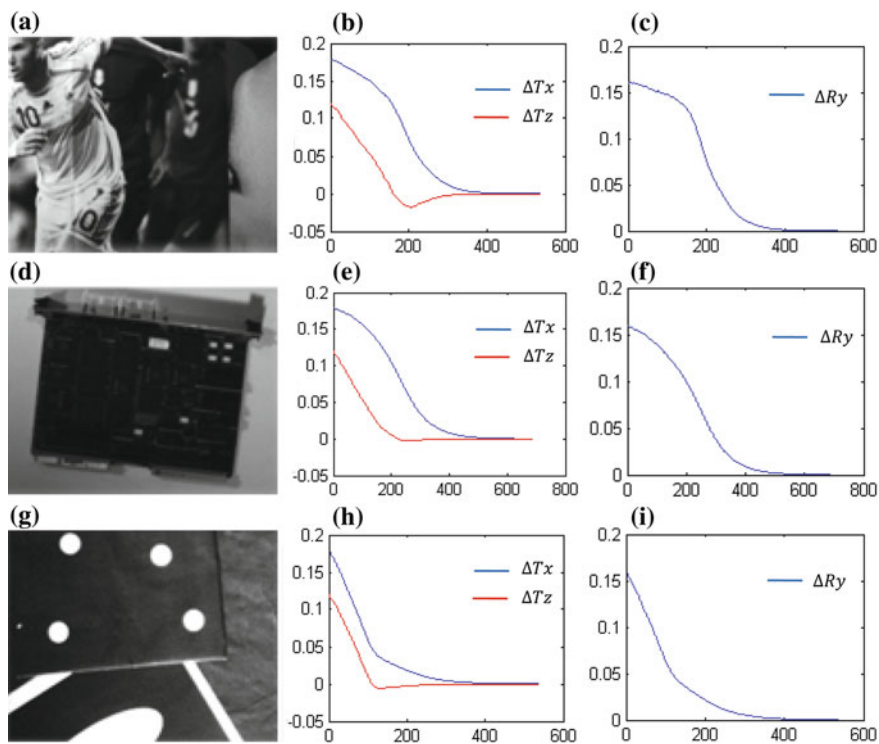


Fig. 12.7 Results of our approach in different cases of scenes. *First column* scenes considered, *second column* translational positioning errors in meter (x axis in frame number), *third column* rotational positioning errors in radian (x axis in frame number)

12.6 Conclusions

In this chapter we focused on the importance of global visual features in visual servoing applications. We found that when the used global feature is the whole image luminance the mobile robot takes so much time to reach its desired pose, therefore we proposed a new approach to achieve fast and real-time visual servoing tasks. This approach is based on a new global feature which is the luminance of a random distribution of image points. To demonstrate the efficiency of this new method our works were, firstly, realized on a virtual platform of VRML then on a real mobile robot. To get the convergence of the robot we have turned the visual servoing problem into an optimization problem. Thus, we have used the control law based on the minimization of a cost function since that ensures the convergence in the case of global visual features.

The new feature has proved to be able to ensure good and fast convergence of the mobile robot even in the case of low textured scenes. As it is global, it does not require any matching nor tracking step and there is no image processing step.

Future works can be intended to verify the robustness of our approach with respect to partial occlusions and large illumination changes.

References

1. Chaumette, F., Hutchinson, S.: Visual servoing and visual tracking. In: Siciliano, B., Khatib, O. (eds.) *Handbook of Robotics*, chapter 24, pp. 563–583. Springer, Berlin (2008)
2. Espiau, B., Chaumette, F., Rives, P.: A new approach to visual servoing in robotics. *IEEE Trans. Robot. Autom.* **8**(3), 313–326 (1992)
3. Chaumette, F., Hutchinson, S.: Visual servo control, part II: Advanced approaches. *IEEE Robot. Autom. Mag.* **14**(1), 109–118 (2007)
4. Collewet, C., Marchand, E., Chaumette, F.: Visual servoing set free from image processing. In: *IEEE International Conference on Robotics and Automation*, pp. 81–86. Pasadena, CA (2008)
5. Chaumette, F., Hutchinson, S.: Application of moment invariants to visual servoing. In: *IEEE International Conference on Robotics and Automation*, pp. 4276–4281. Taiwan (2003)
6. Dame, A., Marchand, E.: Entropy-based visual servoing. In: *IEEE ICRA'09*, pp. 707–713. Kobe (2009)
7. Marchand, E., Collewet, C.: Using image gradient as a visual feature for visual servoing. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5687–5692. Taiwan (2010)
8. Hafez, A., Achar, S., Jawahar, C.: Visual servoing based on gaussian mixture models. In: *IEEE International Conference on Robotics and Automation*, pp. 3225–3230. Pasadena, California (2008)
9. Collewet, C., Marchand, E.: Photometric visual servoing. *IEEE Trans. Robot.* **27**(4), 828–834 (2011)
10. Chaumette, F., Hutchinson, S.: Visual servo control, part I: basic approaches. *IEEE Robot. Autom. Mag.* **13**(4), 82–90 (2006)
11. Marchand, E.: Control camera and light source positions using image gradient information. In: *IEEE International Conference on Robotics and Automation ICRA'07*, pp. 417–422. Roma (2007)
12. Hafez, A., Jawahar, C.: Improvement to the minimization of hybrid error functions for pose alignment. In: *IEEE International Conference on Robotics and Automation Control and Vision*, Singapore (2006)
13. Hafez, A., Jawahar, C.: Visual servoing by optimization of a 2D/3D hybrid objective function. In: *IEEE International Conference on Robotics and Automation*, pp. 1691–1696. Roma (2007)
14. Collewet, C., Marchand, E., Chaumette, F. (ed.) *Luminance: a new visual feature for visual servoing*. In: *Visual Servoing via Advanced Numerical Methods, LNCIS 401*, pp. 71–90. Springer, Berlin (2010)

Chapter 13

Compliance Error Compensation in Robotic-Based Milling

Alexandr Klimchik, Dmitry Bondarenko, Anatol Pashkevich,
Sébastien Briot and Benoît Furet

Abstract This chapter deals with the problem of compliance errors compensation in robotic-based milling. Contrary to previous works that assume that the forces/torques generated by the manufacturing process are constant, the interaction between the milling tool and the workpiece is modeled in details. It takes into account the tool geometry, the number of teeth, the feed rate, the spindle rotation speed and the properties of the material to be processed. Due to high level of the disturbing forces/torques, the developed compensation technique is based on the non-linear stiffness model that allows us to modify the target trajectory taking into account nonlinearities and to avoid the chattering effect. Illustrative example is presented that deals with robotic-based milling of aluminum alloy.

Keywords Industrial robot · Milling · Compliance error compensation · Dynamic machining force model · Non-linear stiffness model

A. Klimchik (✉) · D. Bondarenko · A. Pashkevich
Ecole des Mines de Nantes, 4 rue Alfred-Kastler, 44307 Nantes, France
e-mail: alexandr.klimchik@mines-nantes.fr

D. Bondarenko
e-mail: dmitry.bondarenko@ircyn.ec-nantes.fr

A. Pashkevich
e-mail: anatol.pashkevich@mines-nantes.fr

A. Klimchik · D. Bondarenko · A. Pashkevich · S. Briot · B. Furet
Institut de Recherches en Communications et Cybernétique de Nantes, Nantes, France
e-mail: sebastien.briot@ircyn.ec-nantes.fr

B. Furet
Quai de Tourville, Université de Nantes, 44035 Nantes, France
e-mail: benoit.furet@ircyn.ec-nantes.fr

13.1 Introduction

Currently, robots become more and more popular for a variety of technological processes, including high-speed precision machining. For this process, external loading caused by the machining force is applied on the robot tool. This force is generated by the interaction between the tool mounted on the robot end-effector and the workpiece during the material removal [1]. It is a contact force and it is distributed along the affected area of the tool cutting part. To evaluate the influence and to analyze the robot behavior while machining, the cutting force should be defined either experimentally or using accurate mathematical model.

To evaluate the force caused by interaction between the tool and the workpiece, two approaches can be used. The static approach allows computing the average cutting force without any consideration of dynamic aspect in machining system. This force serves as an external loading of the robot. This approach is widely used in analysis of conventional machining processes using CNC machines [2], where the stiffness is high. In contrast, robots have relatively low structural stiffness. For this reason, in the case of robotic-based machining, an additional source of dynamic displacements of the end-effector with respect to the desired trajectory induced by robot compliance may arise. Such behavior leads to the variable contact between the machining tool and the workpiece. Thus, the generated contact force depends on the current position of the robot end-effector on the trajectory. Consequently, the cutting force cannot be evaluated correctly using the static approach. In this case, the dynamic approach, which will be used in the chapter, is required. It is based on computing of the force at each instant of machining process that defines loading of the robot for the next instant of processing. As a result, the dynamic aspect of robot motion under such variable cutting force can be examined for whole process.

Usually, in the robot-based machining this force causes essential deflections that decrease the quality of the final product. The problem of the robot error compensation can be solved in two ways that differ in degree of modification of the robot control software:

- (a) by *modification of the manipulator model* which better suits to the real manipulator and is used by the robot controller (in simple case, it can be limited by tuning of the nominal manipulator model, but may also involve essential model enhancement by introducing additional parameters, if it is allowed by a robot manufacturer);
- (b) by *modification of the robot control program* that defines the prescribed trajectory in Cartesian space (here, using relevant error model, the input trajectory is generated in such way that under the loading the output trajectory coincides with the desired one, while input trajectory differs from the target one).

Moreover, with regard to the robot-based machining, there is a solution that does not require force/torque measurements or computations [1], where the target trajectory for the robot controller is modified by applying the “mirror” technique. An evident advantage of this technique is its applicability to the compensation of all

types of the robot errors, including geometrical and compliance ones. However, this approach requires carrying out additional preliminary experiments which are quite expensive. So, it is suitable for the large-scale production only. Another compensation methodology has been proposed by Eastwood and Webb [3] that was used for gravitational deflection compensation for hybrid parallel kinematic machines.

This chapter focuses on the modification of control program that is considered to be more realistic in practice. This approach requires also accurate stiffness model of the manipulator. From point of view of stiffness analysis, the external and forces directly influence on the manipulator equilibrium configuration and, accordingly, may modify the stiffness properties. So, they must be undoubtedly taken into account while developing the stiffness model. However, in most of the related works the Cartesian stiffness matrix has been computed for the nominal configuration [4, 5]. Such approach is suitable for the case of small deflections only. For the opposite case, the most important results have been obtained in [6–8], which deal with the stiffness analysis of manipulators under the end-point loading.

Thus, to compensate errors caused by the machining process, it is required to have an accurate stiffness model and precise cutting force model. In contrast to the previous works, the compliance error compensation technique presented in this work is based on the non-linear stiffness model of the manipulator [7] and dynamic model of technological process that generates the cutting force.

13.2 Problem Statement

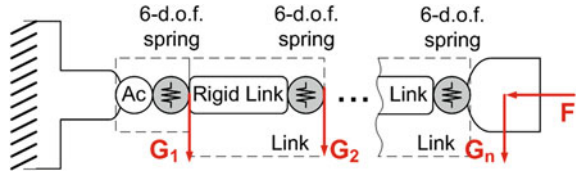
For the *compliance errors*, the compensation technique must rely on two components. The first of them describes distribution of the stiffness properties throughout the workspace and is defined by the stiffness matrix as a function of the joint coordinates. The second component describes the forces/torques acting on the end-effector while the manipulator is performing its machining task (manipulator *loading*).

The *stiffness matrix* required for the compliance errors compensation highly depends on the robot configuration and essentially varies throughout the workspace. From general point of view, full-scale compensation of the compliance errors requires essential revision of the manipulator model embedded in the robot controller. In fact, instead of conventional geometrical model that provides inverse/direct coordinate transformations from the joint to Cartesian spaces and vice versa, here it is necessary to employ the so-called *kinetostatic model* [9]. It is essentially more complicated than the geometrical model and requires rather intensive computations that are presented in Sect. 13.3.

The dynamic behavior of the robot under the loading \mathbf{F} caused by technological process can be described as

$$\mathbf{M}_C \delta \ddot{\mathbf{t}} + \mathbf{C}_C \delta \dot{\mathbf{t}} + \mathbf{K}_C \delta \mathbf{t} = \mathbf{F} \quad (13.1)$$

Fig. 13.1 VJM model of industrial robot with end-point and auxiliary loading



where \mathbf{M}_C is 6×6 mass matrix that represents the global behavior of the robot in terms of natural frequencies, \mathbf{C}_C is 6×6 damping matrix, \mathbf{K}_C is 6×6 Cartesian stiffness matrix of the robot under the external loading \mathbf{F} , $\delta \mathbf{t}$, $\delta \dot{\mathbf{t}}$ and $\delta \ddot{\mathbf{t}}$ are dynamic displacement, velocity and acceleration of the tool end-point in a current moment respectively [10].

In general, the cutting force F_c has a *nonlinear* nature and depends on many factors such as cutting conditions, properties of workpiece material and tool cutting part, etc [11]. But, for given tool/workpiece combination, the force F_c could be approximated as a function of an uncut chip thickness h , which represents the desired thickness to cut at each instant of machining.

Hence, to reduce the errors caused by the cutting forces in the robotic-based machining it is required to obtain an accurate elastostatic model of the robot and elastodynamic model of the machining process. These problems are addressed in the following sections taking into account some particularities of the considered application (robotic-based milling).

13.3 Manipulator Model

13.3.1 Elastostatic Model

Elastostatic model of a serial robot is usually defined by its Cartesian stiffness matrix, which should be computed in the neighborhood of loaded configuration. Let us propose numerical technique for computing static equilibrium configuration for a general type of serial manipulator. Such manipulator may be approximated as a set of rigid links and virtual joints, which take into account elastostatic properties (Fig. 13.1). Since the link weight of serial robots is not negligible, it is reasonable to decompose it into two parts (based on the link mass centre) and apply them to the both ends of the link. All this loadings will be aggregated in a vector $\mathbf{G} = [\mathbf{G}_1 \dots \mathbf{G}_n]$, where \mathbf{G}_i is the loading applied to the i th node-point. Besides, it is assumed that the external loading \mathbf{F} (caused by the interaction of the tool and the workpiece) is applied to the robot end-effector.

Following the principle of virtual work, the work of external forces \mathbf{G} , \mathbf{F} is equal to the work of internal forces $\boldsymbol{\tau}_\theta$ caused by displacement of the virtual springs $\delta \boldsymbol{\theta}$

$$\sum_{j=1}^n \left(\mathbf{G}_j^T \cdot \delta \mathbf{t}_j \right) + \mathbf{F}^T \cdot \delta \mathbf{t} = \boldsymbol{\tau}_\theta^T \cdot \delta \boldsymbol{\theta} \quad (13.2)$$

where the virtual displacements $\delta \mathbf{t}_j$ can be computed from the linearized geometrical model derived from $\delta \mathbf{t}_j = \mathbf{J}_\theta^{(j)} \delta \boldsymbol{\theta}$, $j = 1 \dots n$, which includes the Jacobian matrices $\mathbf{J}_\theta^{(j)} = \partial \mathbf{g}_j(\mathbf{q}, \boldsymbol{\theta}) / \partial \boldsymbol{\theta}$ with respect to the virtual joint coordinates.

So, expression (13.2) can be rewritten as

$$\sum_{j=1}^n \left(\mathbf{G}_j^T \cdot \mathbf{J}_\theta^{(j)} \cdot \delta \boldsymbol{\theta} \right) + \left(\mathbf{F}^T \cdot \mathbf{J}_\theta^{(n)} \cdot \delta \boldsymbol{\theta} \right) = \boldsymbol{\tau}_\theta^T \cdot \delta \boldsymbol{\theta} \quad (13.3)$$

which has to be satisfied for any variation of $\delta \boldsymbol{\theta}$. It means that the terms regrouping the variables $\delta \boldsymbol{\theta}$ have the coefficients equal to zero. Hence the force balance equations can be written as

$$\boldsymbol{\tau}_\theta = \sum_{j=1}^n \mathbf{J}_\theta^{(j)T} \cdot \mathbf{G}_j + \mathbf{J}_\theta^{(n)T} \cdot \mathbf{F} \quad (13.4)$$

These equations can be re-written in block-matrix form as

$$\boldsymbol{\tau}_\theta = \mathbf{J}_\theta^{(G)T} \cdot \mathbf{G} + \mathbf{J}_\theta^{(F)T} \cdot \mathbf{F} \quad (13.5)$$

where $\mathbf{J}_\theta^{(F)} = \mathbf{J}_\theta^{(n)}$, $\mathbf{J}_\theta^{(G)} = \left[\mathbf{J}_\theta^{(1)T} \dots \mathbf{J}_\theta^{(n)T} \right]^T$, $\mathbf{G} = \left[\mathbf{G}_1^T \dots \mathbf{G}_n^T \right]^T$. Finally, taking into account the virtual spring reaction $\boldsymbol{\tau}_\theta = \mathbf{K}_\theta \cdot \boldsymbol{\theta}$, where $\mathbf{K}_\theta = \text{diag}(\mathbf{K}_{\theta_1}, \dots, \mathbf{K}_{\theta_n})$, the desired static equilibrium equations can be presented as

$$\mathbf{J}_\theta^{(G)T} \cdot \mathbf{G} + \mathbf{J}_\theta^{(F)T} \cdot \mathbf{F} = \mathbf{K}_\theta \cdot \boldsymbol{\theta} \quad (13.6)$$

To obtain a relation between the external loading \mathbf{F} and internal coordinates of the kinematic chain $\boldsymbol{\theta}$ corresponding to the static equilibrium, Eq. (13.6) should be solved either for different given values of \mathbf{F} or for different given values of \mathbf{t} . Let us solve the static equilibrium equations with respect to the manipulator configuration $\boldsymbol{\theta}$ and the external loading \mathbf{F} for given end-effector position $\mathbf{t} = \mathbf{g}(\boldsymbol{\theta})$ and the function of auxiliary-loadings $\mathbf{G}(\boldsymbol{\theta})$

$$\mathbf{K}_\theta \cdot \boldsymbol{\theta} = \mathbf{J}_\theta^{(G)T} \mathbf{G} + \mathbf{J}_\theta^{(F)T} \mathbf{F}; \mathbf{t} = \mathbf{g}(\boldsymbol{\theta}); \mathbf{G} = \mathbf{G}(\boldsymbol{\theta}) \quad (13.7)$$

where the unknown variables are $(\boldsymbol{\theta}, \mathbf{F})$.

Since usually this system has no analytical solution, iterative numerical technique can be applied. So, the kinematic equations may be linearized in the neighborhood of the current configuration $\boldsymbol{\theta}_i$

$$\mathbf{t}_{i+1} = \mathbf{g}(\boldsymbol{\theta}_i) + \mathbf{J}_\theta^{(F)}(\boldsymbol{\theta}_i) \cdot (\boldsymbol{\theta}_{i+1} - \boldsymbol{\theta}_i); \quad (13.8)$$

where the subscript ‘ i ’ indicates the iteration number and the changes in Jacobians $\mathbf{J}_\theta^{(G)}$, $\mathbf{J}_\theta^{(F)}$ and the auxiliary loadings \mathbf{G} are assumed to be negligible from iteration to iteration. Correspondingly, the static equilibrium equations in the neighborhood of $\boldsymbol{\theta}_i$ may be rewritten as

$$\mathbf{J}_\theta^{(G)T} \cdot \mathbf{G} + \mathbf{J}_\theta^{(F)T} \cdot \mathbf{F}_{i+1} = \mathbf{K}_\theta \cdot \boldsymbol{\theta}_{i+1} \quad (13.9)$$

Thus, combining (13.8), (13.9) and expression for $\boldsymbol{\theta} = \mathbf{K}_\theta^{-1}(\mathbf{J}_\theta^{(G)T} \cdot \mathbf{G} + \mathbf{J}_\theta^{(F)T} \cdot \mathbf{F})$, the unknown variables \mathbf{F} and $\boldsymbol{\theta}$ can be computed using following iterative scheme

$$\begin{aligned} \mathbf{F}_{i+1} &= \left(\mathbf{J}_\theta^{(F)} \cdot \mathbf{K}_\theta^{-1} \cdot \mathbf{J}_\theta^{(F)T} \right)^{-1} \left(\mathbf{t}_{i+1} - \mathbf{g}(\boldsymbol{\theta}_i) + \mathbf{J}_\theta^{(F)} \boldsymbol{\theta}_i - \mathbf{J}_\theta^{(F)} \mathbf{K}_\theta^{-1} \mathbf{J}_\theta^{(G)T} \mathbf{G}_i \right) \\ \boldsymbol{\theta}_{i+1} &= \mathbf{K}_\theta^{-1} \left(\mathbf{J}_\theta^{(G)T} \cdot \mathbf{G}_i + \mathbf{J}_\theta^{(F)T} \cdot \mathbf{F}_{i+1} \right) \end{aligned} \quad (13.10)$$

The proposed algorithm allows us to compute the static equilibrium configuration for the serial robot under external loadings applied to any point of the manipulator and the loading from the technological process.

13.3.2 Stiffness Matrix

In order to obtain the Cartesian stiffness matrix, let us linearize the force-deflection relation in the neighborhood of the equilibrium. Following this approach, two equilibriums that correspond to the manipulator state variables $(\mathbf{F}, \boldsymbol{\theta}, \mathbf{t})$ and $(\mathbf{F} + \delta\mathbf{F}, \boldsymbol{\theta} + \delta\boldsymbol{\theta}, \mathbf{t} + \delta\mathbf{t})$ should be considered simultaneously. Here, notations $\delta\mathbf{F}$, $\delta\mathbf{t}$ define small increments of the external loading and relevant displacement of the end-point. Finally, the static equilibrium equations may be written as

$$\mathbf{t} = \mathbf{g}(\boldsymbol{\theta}); \quad \mathbf{K}_\theta \cdot \boldsymbol{\theta} = \mathbf{J}_\theta^{(G)T} \cdot \mathbf{G} + \mathbf{J}_\theta^{(F)T} \cdot \mathbf{F} \quad (13.11)$$

and

$$\begin{aligned} \mathbf{t} + \delta\mathbf{t} &= \mathbf{g}(\boldsymbol{\theta} + \delta\boldsymbol{\theta}) \\ \mathbf{K}_\theta \cdot (\boldsymbol{\theta} + \delta\boldsymbol{\theta}) &= \left(\mathbf{J}_\theta^{(G)} + \delta\mathbf{J}_\theta^{(G)} \right)^T \cdot (\mathbf{G} + \delta\mathbf{G}) + \left(\mathbf{J}_\theta^{(F)} + \delta\mathbf{J}_\theta^{(F)} \right)^T \cdot (\mathbf{F} + \delta\mathbf{F}) \end{aligned} \quad (13.12)$$

where \mathbf{t} , \mathbf{F} , \mathbf{G} , \mathbf{K}_θ , $\boldsymbol{\theta}$ are assumed to be known.

After linearization of the function $\mathbf{g}(\boldsymbol{\theta})$ in the neighborhood of the loaded equilibrium, the system (13.11), (13.12) is reduced to equations

$$\begin{aligned}\delta \mathbf{t} &= \mathbf{J}_\theta^{(F)} \delta \boldsymbol{\theta} \\ \mathbf{K}_\theta \cdot \delta \boldsymbol{\theta} &= \delta \mathbf{J}_\theta^{(G)} \mathbf{G} + \mathbf{J}_\theta^{(G)} \delta \mathbf{G} + \delta \mathbf{J}_\theta^{(F)} \mathbf{F} + \mathbf{J}_\theta^{(F)} \delta \mathbf{F}\end{aligned}\quad (13.13)$$

which defines the desired linear relations between $\delta \mathbf{t}$ and $\delta \mathbf{F}$. In this system, small variations of Jacobians may be expressed via the second order derivatives $\delta \mathbf{J}_\theta^{(F)} = \mathbf{H}_{\theta\theta}^{(F)} \cdot \delta \boldsymbol{\theta}$, $\delta \mathbf{J}_\theta^{(G)} = \mathbf{H}_{\theta\theta}^{(G)} \cdot \delta \boldsymbol{\theta}$, where

$$\mathbf{H}_{\theta\theta}^{(G)} = \sum_{j=1}^n \partial^2 \mathbf{g}_j^T \mathbf{G}_j / \partial \boldsymbol{\theta}^2; \quad \mathbf{H}_{\theta\theta}^{(F)} = \partial^2 \mathbf{g}^T \mathbf{F} / \partial \boldsymbol{\theta}^2 \quad (13.14)$$

Also, the auxiliary loading \mathbf{G} may be computed via the first order derivatives as $\delta \mathbf{G} = \partial \mathbf{G} / \partial \boldsymbol{\theta} \cdot \delta \boldsymbol{\theta}$

Further, let us introduce additional notation

$$\mathbf{H}_{\theta\theta} = \mathbf{H}_{\theta\theta}^{(F)} + \mathbf{H}_{\theta\theta}^{(G)} + \mathbf{J}_\theta^{(G)T} \cdot \partial \mathbf{G} / \partial \boldsymbol{\theta} \quad (13.15)$$

which allows us to present system (13.13) in the form

$$\begin{bmatrix} \delta \mathbf{t} \\ \mathbf{0} \end{bmatrix} = \begin{bmatrix} \mathbf{0} & \mathbf{J}_\theta^{(F)} \\ \mathbf{J}_\theta^{(F)T} & -\mathbf{K}_\theta + \mathbf{H}_{\theta\theta} \end{bmatrix} \cdot \begin{bmatrix} \delta \mathbf{F} \\ \delta \boldsymbol{\theta} \end{bmatrix} \quad (13.16)$$

So, the desired Cartesian stiffness matrices \mathbf{K}_C can be computed as

$$\mathbf{K}_C = \left(\mathbf{J}_\theta^{(F)} (\mathbf{K}_\theta - \mathbf{H}_{\theta\theta})^{-1} \mathbf{J}_\theta^{(F)T} \right)^{-1} \quad (13.17)$$

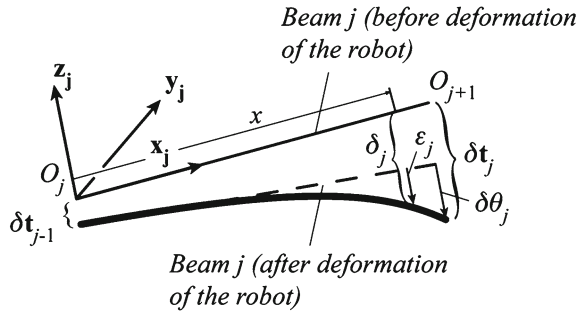
Below, this expression will be used for computing of the elastostatic deflections of the robotic manipulator.

13.3.3 Reduced Mass Matrix

To evaluate the dynamic behavior of the robot under the loading, in addition to the Cartesian stiffness matrix \mathbf{K}_C it is required to define the Cartesian mass matrix \mathbf{M}_C . This mass matrix has the same dimension as \mathbf{K}_C and can be obtained using some model reduction techniques. Comprehensive analysis and definition of this matrix have been proposed in [10]. Here, let us summarize the main results that will be used further.

To reduce the mass matrix dimension, model reduction techniques are applied for decreasing the size of the link mass matrices and also for the robot total mass matrix. Two main ways can be followed to reduce the size of the link mass matrices. The first one consists in discretizing the beam j into p_j rigid links and springs and

Fig. 13.2 Displacements and elastic deformations of a beam



to express their displacements as a function of the beam extremity displacements. However, such numerical method must be repeated for each link and, thus, increases the size of the algorithm and decreases its efficiency. As a result, it is preferred to use the following procedure which allows analytical expressions to be obtained for the reduced link mass matrices.

Let us consider the link j , modeled as a beam (Fig. 13.2). At this beam is attached a local frame represented by the vectors \mathbf{x}_j , \mathbf{y}_j and \mathbf{z}_j . Before any deformation of the system, the beam j is linked to beams $(j - 1)$ and $(j + 1)$ at points O_j and O_{j+1} , respectively (Fig. 13.2). After deformation of the robot, the beam extremity located at O_j is displaced from $\delta \mathbf{t}_{j-1} = [\delta t_{j-1}^1, \delta t_{j-1}^2, \dots, \delta t_{j-1}^6]^T$ and the one located at O_{j+1} is displaced from $\delta \mathbf{t}_j = [\delta t_j^1, \delta t_j^2, \dots, \delta t_j^6]^T$, where the three first components of each vector correspond to the translational displacements along local \mathbf{x}_j , \mathbf{y}_j and \mathbf{z}_j axes, respectively, and the three last components to the rotational displacements along the same axes.

The general formula for the kinetic energy of an elastic Bernoulli beam is equal to:

$$T_j = 1/2 \int_0^{L_j} \rho_j \dot{\delta}_j^T \mathbf{Q}_j \dot{\delta}_j dx; \quad \mathbf{Q}_j = \text{diag} (A_j, A_j, A_j I_j^p, I_j^y, I_j^z) \quad (13.18)$$

In this expression, $\dot{\delta}_j$ represents the velocity of the beam cross-section located at position x from the local reference frame (Fig. 13.2), L_j is the length of the beam j , ρ_j the mass density at cross-section x , A_j its area, I_j^p its torsional constant and I_j^y , I_j^z , the quadratic momentums along y_j and z_j , respectively.

For the l th natural mode ω_l , the kinetic energy can be rewritten as:

$$T_{jl} = 1/2 \omega_l^2 \cos^2 (\omega_l t + \varphi_l) \int_0^{L_j} \rho_j \delta_j^T \mathbf{Q}_j \delta_j dx \quad (13.19)$$

δ_j being the amplitude of the displacement of the beam cross-section located at position x from the local reference frame (Fig. 13.2).

In the Rayleigh-Ritz approximation, considering that the deformations due to the natural vibrations are similar to those obtained when an external load is applied at the robot end-effector only, each link of the structure will deform due to the stresses transmitted through the robot joints at points O_j . As a result, the deformations $\boldsymbol{\varepsilon}_j$ of the beam cross-section can be approximated by the deformations of a tip-loaded beam

$$\boldsymbol{\varepsilon}_j = \text{diag} (f_j, g_j, g_j, f_j, h_j, h_j) \boldsymbol{\delta}\boldsymbol{\theta}_j \quad (13.20)$$

where $\boldsymbol{\delta}\boldsymbol{\theta}_j = \boldsymbol{\varepsilon}_j (x = L_j)$ represents the deformation of the beam at its tip and

$$f_j (x) = x/L_j, \quad g_j (x) = 0.5x^2 (3L_j - x)/L_j^3, \quad h_j (x) = 2x (L_j - 0.5x)/L_j^2 \quad (13.21)$$

As a result, the global displacement $\boldsymbol{\delta}_j$ of the beam cross-section at x can be expressed as a sum of two terms:

$$\boldsymbol{\delta}_j = \begin{bmatrix} \mathbf{I}_3 & \mathbf{D}_{(\times)} \\ \mathbf{0}_3 & \mathbf{I}_3 \end{bmatrix} \boldsymbol{\delta}\mathbf{t}_{j-1} + \boldsymbol{\varepsilon}_j, \quad \text{with } D_{(\times)} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -x \\ 0 & x & 0 \end{bmatrix} \quad (13.22)$$

In this sum, the left terms corresponds to the displacement of the undeformed beam due to the displacement of the node located at O_j .

Introducing (13.20–13.22) into (13.9) leads to the following equation:

$$T_{jl} = 1/2\omega_l^2 \cos^2 (\omega_l t + \varphi_l) \left(\begin{bmatrix} \boldsymbol{\delta}\mathbf{t}_{j-1}^T & \boldsymbol{\delta}\mathbf{t}_j^T \end{bmatrix} \mathbf{M}_j^{\text{red}} \begin{bmatrix} \boldsymbol{\delta}\mathbf{t}_{j-1} \\ \boldsymbol{\delta}\mathbf{t}_j \end{bmatrix} \right) \quad (13.23)$$

where the expressions of each components of matrix $\mathbf{M}_j^{\text{red}}$ are given in [10].

Using these results, the total kinetic energy of the system for the l th node is:

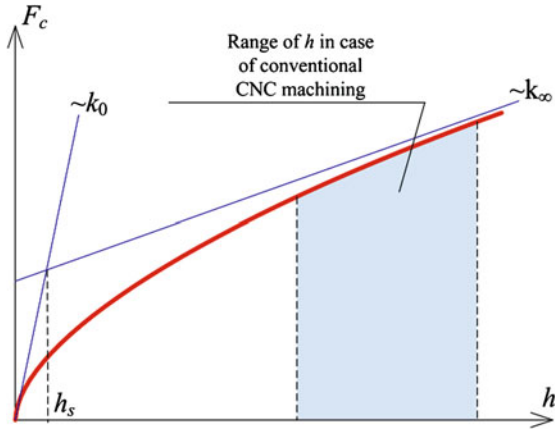
$$T_l = \sum_j T_{jl} = 1/2\omega_l^2 \cos^2 (\omega_l t + \varphi_l) \boldsymbol{\delta}\mathbf{t}^T \mathbf{M}_{tot} \boldsymbol{\delta}\mathbf{t} \quad (13.24)$$

with $\mathbf{M}_{tot} = \text{diag} (\mathbf{M}_1^{\text{red}}, \dots, \mathbf{M}_n^{\text{red}})$ and $\boldsymbol{\delta}\mathbf{t}^T = [\boldsymbol{\delta}\mathbf{t}_0^T, \boldsymbol{\delta}\mathbf{t}_1^T, \dots, \boldsymbol{\delta}\mathbf{t}_{n-1}^T, \boldsymbol{\delta}\mathbf{t}_n^T]$

Then, assuming that the first natural modes of vibrations, i.e. the modes that have the most energy, lead to deformations that are close to the static deformations of the robot under a load applied on the end-effector, the mass matrix can be recomputed into the Cartesian coordinates associated with the tool end-point using the Jacobian matrix \mathbf{J}_θ defined at expression (13.3) (which depend on the robot configuration \mathbf{q} and computed with respect to virtual joint coordinates $\boldsymbol{\theta}$) using following expression

$$\mathbf{M}_C = \mathbf{J}_\theta^T \mathbf{M}_\theta \mathbf{J}_\theta \quad (13.25)$$

Fig. 13.3 Fractional cutting force model $F_c(h)$



Thus, using expressions (13.25), it is possible to compute the reduced mass matrix \mathbf{M}_C for a given robot configuration \mathbf{q} . The performances of this model reduction are shown in [10].

13.4 Machining Process

Let us obtain the model of the cutting force which depends on the relative position of the tool with respect to the workpiece at each instant of machining. As follows from previous works [12], for the known chip thickness h , the cutting force F_c can be expressed as

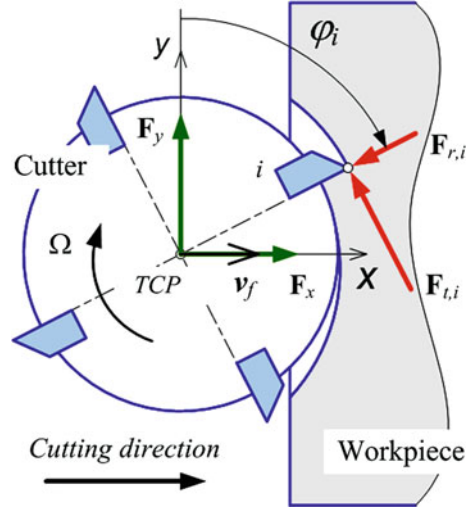
$$F_c(h) = k_0 \frac{h/h_s + r (h/h_s)^2}{1 + h/h_s} a_p, \quad h \geq 0 \quad (13.26)$$

where a_p is a depth of cut, $r = k_\infty/k_0 < 1$ depends on the parameters k_∞, k_0 that define the so called stiffness of the cutting process for large and small chip thickness h respectively (Fig. 13.3) and h_s is a specific chip thickness, which depends on the current state of the tool cutting edge. The parameters k_0, h_s, r are evaluated experimentally for a given combination of tool/working material. To take into account the possible loss of contact between the tool and the workpiece, the above expression should be supplemented by the case of $h < 0$ as

$$F_c(h) = 0, \quad \text{if } h < 0 \quad (13.27)$$

For the multi-edge tool the machining surface is formed by means of several edges simultaneously. The number of working edges varies during machining and depends on the width of cut. For this reason, the total force F_c of such interaction is a superposition of forces $F_{c,i}$ generated by each tool edge i , which are currently in the

Fig. 13.4 Forces of tool/workpiece interaction



contact with the workpiece. Besides, the contact force $F_{c,i}$ can be decomposed by its *radial* $F_{r,i}$ and *tangential* $F_{t,i}$ components (Fig. 13.4). In accordance with Merchant's model [13], the *t*-component of cutting force $F_{t,i}$ can be computed with the Eq. (13.26). The *r*-component $F_{r,i}$ is related with $F_{t,i}$ by following expression [14]

$$F_{r,i} = k_r F_{t,i} \quad (13.28)$$

where the ratio factor k_r depends on the given tool/workpiece characteristics.

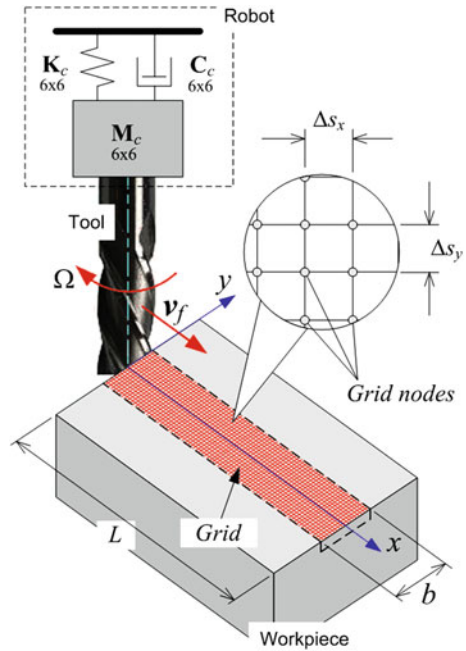
It should be mentioned that in robotic machining it is more suitable to operate with forces expressed in the robot tool frame $\{x, y, z\}$. Then, the corresponding components F_x, F_y (Fig. 13.4) of the cutting force F_c can be expressed as follows

$$\begin{aligned} F_x &= \sum_{i=1}^{n_z} -F_{r,i} \cos \varphi_i + \sum_{i=1}^{n_z} F_{t,i} \sin \varphi_i \\ F_y &= \sum_{i=1}^{n_z} F_{r,i} \sin \varphi_i + \sum_{i=1}^{n_z} F_{t,i} \cos \varphi_i \end{aligned} \quad (13.29)$$

where n_z is the number of currently working cutting edges, φ_i is the angular position of the i th cutting edge (the cutting force in z direction F_z is negligible here). So, the vector of external loading of the robot due to the machining process can be composed in the frame $\{x, y, z\}$ using the defined components F_x, F_y as $\mathbf{F} = [F_x, F_y, 0, 0, 0, 0]^T$.

It should be stressed that the cutting force components $F_{r,i}, F_{t,i}$ mentioned in Eqs. (13.26), (13.28) are computed for the given chip thickness h_i , which should be also evaluated. Let us define model for h_i using mechanical approach. Then the chip thickness h_i removed by i th tooth depends on the angular position φ_i of this tooth and it can be evaluated using to the geometrical distance between the position of the

Fig. 13.5 Meshing of the workpiece area



given tooth i and the current machining profile (Fig. 13.4). It should be mentioned, that the main issue here is to follow the current relative position between the i th tooth and the working material or to define whether the i th tooth is involved in cutting for given instant of process. Because of the robot dynamic behavior and the regenerative mechanism of surface formation [15] this problem cannot be solved directly using kinematic relations. In this case it is reasonable to introduce a special rectangular grid, which decomposes the workpiece area into segments and allows tracking the tool/workpiece interaction and the formation of the machining profile (Fig. 13.5).

Here, Steps $\Delta s_x, \Delta s_y$ between grid nodes are constant and depend on the tool geometry, cutting condition and time discretization $\Delta \tau$. Each node j ($j = \overline{1, N_w}, N_w$ is the number of nodes) of the grid can be marked as “1” or “0”: “1” corresponds to nodes situated in the workpiece area with material (rose nodes in Fig. 13.6), “0” corresponds to nodes situated in workpiece area that was cut away (white nodes in Fig. 13.6).

In order to define the number of currently cut nodes by the i th tooth, the previous instant of machining process should be considered. Let us define A_i as an amount of working material that is currently cut away by the i th tooth (Fig. 13.6). So, if node j marked as “1” is located inside the marked sector (green nodes in Fig. 13.6), it changes to “0” and A_i is increasing by $\Delta s_x \Delta s_y$. Analyzing all potential nodes and computing A_i , the chip thickness h_i , removed at given instant of the process by the i th tooth, can be estimated by $h_i = A_i / R \Delta \alpha_i, \quad i = \overline{1, N_z}$. The angle $\Delta \varphi_i$ determines

Fig. 13.6 Evaluating the tool/workpiece intersection A_i and computing the corresponding chip thickness h_i

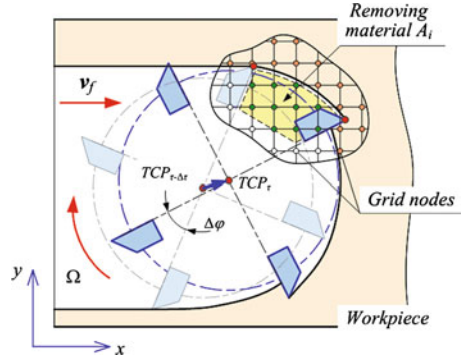
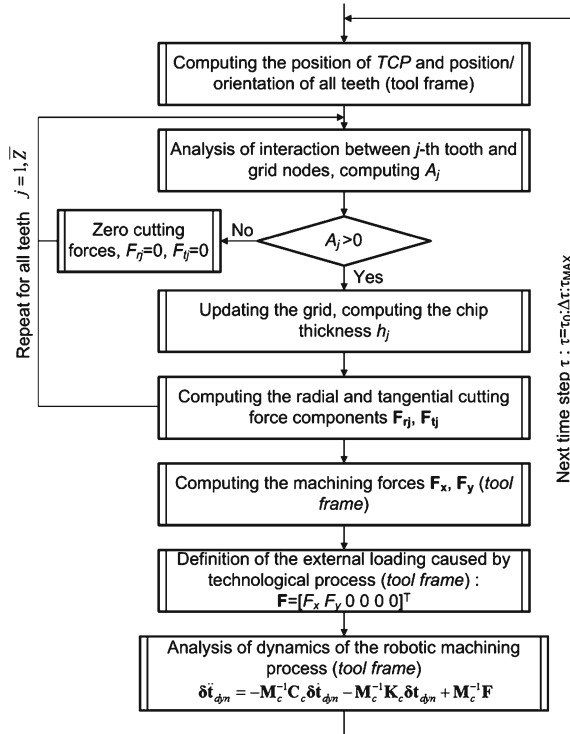


Fig. 13.7 Algorithm for numerical simulation of robotic machining process dynamics



the current angular position of the i th tooth regarding to its position at the instant $\tau - \Delta\tau$ and referred to the position of TCP at $\tau - \Delta\tau$.

Described mechanism of chip formation and the machining force model (13.26) allow computing the dynamic behavior of the robotic machining process where models of robot inertia and stiffness are discussed in the Sect. 13.3 of the chapter. The detailed algorithm that is used in numerical analysis is presented in Fig. 13.7, where the analysis of the robot dynamics is performed in the tool frame with respect to

the dynamic displacement of the tool δt_{dyn} fixed on the robot end-effector around its position on the trajectory.

13.5 Compliance Error Compensation Technique

In industrial robotic controllers, the manipulator motions are usually generated using the inverse kinematic model that allows us to compute the input signals for actuators ρ_0 corresponding to the desired end-effector location \mathbf{t}_0 , which is assigned assuming that the compliance errors are negligible. However, if the external loading \mathbf{F} is essential, the kinematic control becomes non-applicable because of changes in the end-effector location. It can be computed from the non-linear compliance model as

$$\mathbf{t}_F = f^{-1}(\mathbf{F} | \mathbf{t}_0) \quad (13.30)$$

where the subscripts 'F' and '0' refer to the loaded and unloaded modes respectively, and '|' separates arguments and parameters of the function $f(\cdot)$. Some details concerning this function are given in our previous publication [7].

To compensate this undeterred end-effector displacement from \mathbf{t}_0 to \mathbf{t}_F , the target point should be modified in such a way that, under the loading \mathbf{F} , the end-platform is located in the desired point \mathbf{t}_0 . This requirement can be expressed using the stiffness model in the following way

$$\mathbf{F} = f(\mathbf{t}_0 | \mathbf{t}_0^{(F)}) \quad (13.31)$$

where $\mathbf{t}_0^{(F)}$ denotes the modified target location. Hence, the problem is reduced to the solution of the nonlinear Eq. (13.31) for $\mathbf{t}_0^{(F)}$, while \mathbf{F} and \mathbf{t}_0 are assumed to be given. It is worth mentioning that this equation completely differs from the equation $\mathbf{F} = f(\mathbf{t} | \mathbf{t}_0)$, where the unknown variable is \mathbf{t} . It means that here the compliance model does not allow us to compute the modified target point $\mathbf{t}_0^{(F)}$ straightforwardly, while the linear compensation technique directly operates with Cartesian compliance matrix [16].

To solve Eq. (13.31) for $\mathbf{t}_0^{(F)}$, similar numerical technique can be applied. It yields the following iterative scheme

$$\mathbf{t}_0^{(F)'} = \mathbf{t}_0^{(F)} + \alpha \cdot (\mathbf{t}_0 - f^{-1}(\mathbf{F} | \mathbf{t}_0^{(F)})) \quad (13.32)$$

where the prime corresponds to the next iteration, $\alpha \in (0, 1)$ is the scalar parameter ensuring the convergence. More detailed presentation of the developed iterative routines is given in Fig. 13.8.

Hence, using the proposed computational techniques, it is possible to compensate a main part compliance errors by proper adjusting the reference trajectory that is used as an input for robotic controller. In this case, the control is based on the

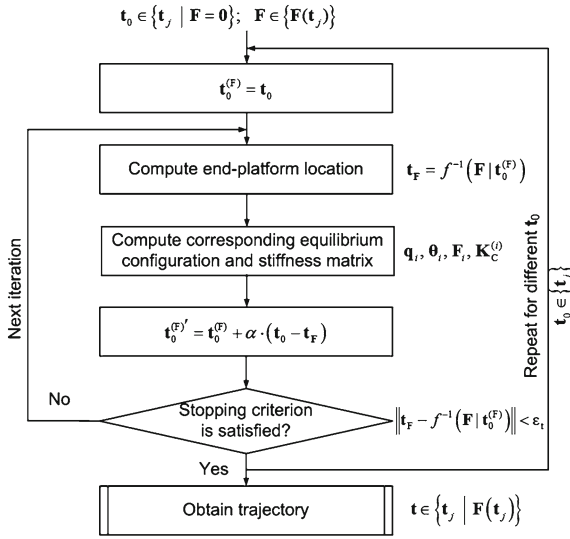
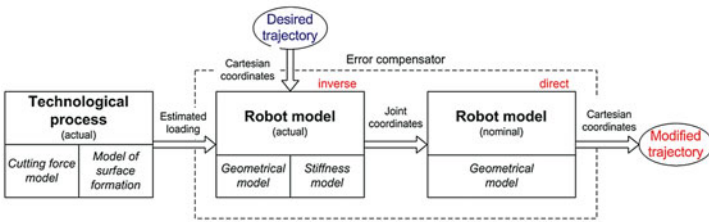


Fig. 13.8 Procedure for compensation of compliance errors

(a)



(b)

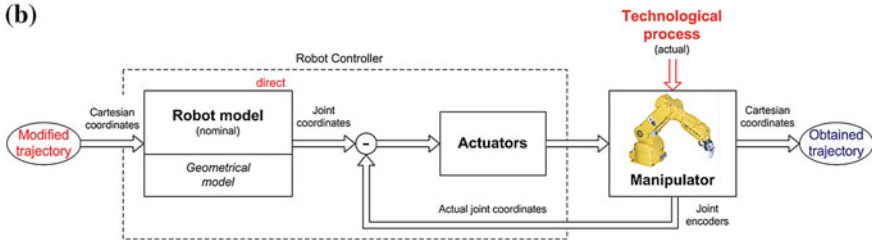
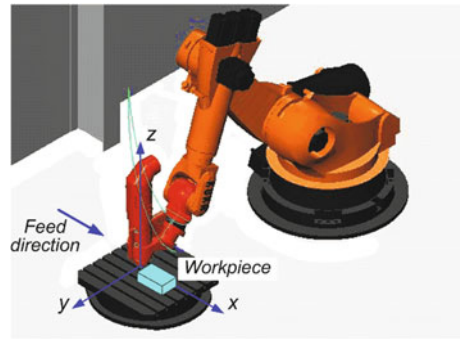


Fig. 13.9 Implementation of compliance error compensation technique

inverse kinetostatic model (instead of kinematic one) that takes into account both the manipulator geometry and elastic properties of its links and joints. Implementation of developed compliance error compensation technique presented in Fig. 13.9.

Table 13.1 Initial data for robotic-based milling

<i>Joint coordinates (deg)</i>					
q_1	q_2	q_3	q_4	q_5	q_6
90	-50	120	180	25	180
<i>Joint compliances (rad/N m)*10^{-6}</i>					
k_1	k_2	k_3	k_4	k_5	k_6
0.26	0.15	0.26	1.79	1.52	2.13
<i>Link masses (kg)</i>					
m_1	m_2	m_3	m_4	m_5	m_6
336.8	259.4	85.2	54.5	36.3	18.2

Fig. 13.10 Starting pose of the KUKA KR270 robot to perform the operation of milling

13.6 Experimental Verification

The developed compliance error compensation technique has been verified experimentally for robotic milling with the KUKA KR270 robot along a simple trajectory in aluminum workpiece. It is assumed that at the beginning of the technological process the robot is in the configuration \mathbf{q} (see Table 13.1, Fig. 13.10). The parameters of the stiffness model for the considered robot have been identified in [17] and are presented in Table 13.1. Link masses required for the mass matrix of the robot are presented also in Table 13.1.

For the milling, the cutter with the external diameter $D = 20$ mm and four teeth ($N_z = 4$) distributed uniformly over the tool is used. For the given combination of the tool and the workpiece material the following parameters correspond to the cutting force model defined in (13.26): $k_0 = 5 \times 10^6$ N/m, $h_s = 1.8 \times 10^{-5}$ m, $r = 0.1$, $k_r = 0.3$.

Taking into account that the workpiece has a straight borders let us assume that at the instant $t = 0$ one of the teeth of the tool is in contact with the workpiece material as it is shown in the Fig. 13.11. It is also assumed that the machining process is performing with the constant feed rate $v_f = 4$ m/min (applied in x -direction of the robot tool frame) and the constant spindle rotation $\Omega = 8000$ rpm along the straight line of 80 mm. Experimental verification and numerical simulation of the described

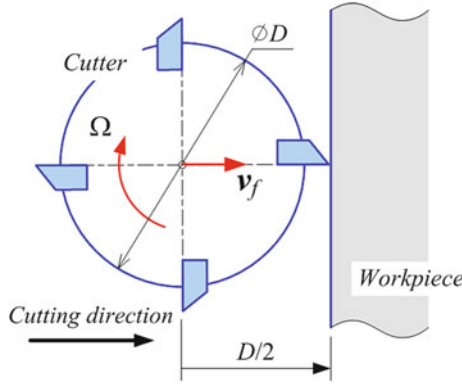


Fig. 13.11 Starting relative position of the tool with respect to the workpiece

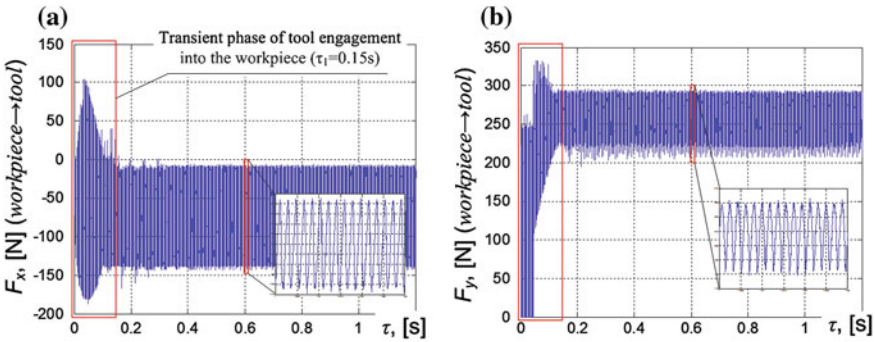


Fig. 13.12 Variation of machining force components F_x (a) and F_y (b) for whole milling process

case of the milling process with KUKA KR-270 robot using the algorithm shown in Fig. 13.7 allows us to trace the evolution of machining force x, y-components for the whole process (Fig. 13.12). The corresponding dynamic displacement of the tool around its current position on the trajectory is shown in Fig. 13.13.

In accordance with the obtained results the system robot/machining process realize complex vibratory motion. The high frequency component of this motion (about 700 Hz, Fig. 13.12) is related to the spindle rotation and the number of tool teeth N_z . In certain cases such behavior can excites the dynamics of the robot (natural modes) but this study remains out the frame of the presented chapter. On the contrary, the low frequency component of robot/tool motion (about 7 Hz, Fig. 13.13), especially in the y-direction (that is perpendicular to the applied feed) influences directly the quality of final product. Such motion is related to the robot compliance and it can be compensated using the error compensation technique described in the chapter. Hence, let us form the modified trajectory based on the dynamic displacement of the robot end-effector in the y-direction (Fig. 13.14).

Fig. 13.13 Evolution of the tool dynamic displacement δt_{dyn} that is composed from x_{TCP} and y_{TCP} components

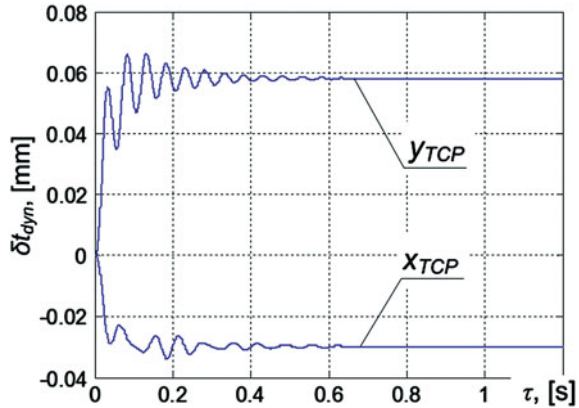
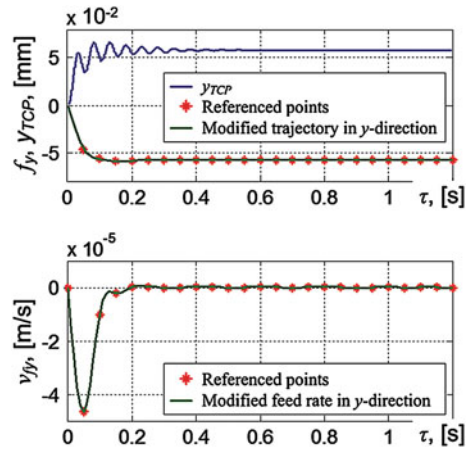


Fig. 13.14 Modified trajectory f_y and corresponding feed rate v_{fy} in y-direction, computed based on the original dynamic displacement of the tool δt_{dyn}



It should be stressed that the time step between referenced points of this modified trajectory is limited with the characteristics of the controller used in the robot (in the presented case this step is chosen 0.05 s). The corresponding feed rate v_{fy} for the modified trajectory has been computed. So, this new data (feed f_y and feed rate v_{fy}) with the data defined in the beginning of this section allow us to compensate the trajectory error during machining caused by the robot compliance. The resulted compensated trajectory in the y-direction (in time domain) is presented in Fig. 13.15.

It should be noted that the part of the trajectory while machining tool is engaging into the workpiece does not have effect on the quality of final product (surface). During this stage the contact area between the tool and the workpiece is increasing progressively. Hence, at each instant of processing the cutter corrects the machining profile and eliminates trajectory errors produced during all previous instants. On the contrary, during the stage of machining with the fully engaged tool the trajectory in x, y-directions define directly the final machining profile and this part of trajectory

Fig. 13.15 Evolution of the dynamic displacement obtained after involving the error compensation technique into the analysis of robotic milling process

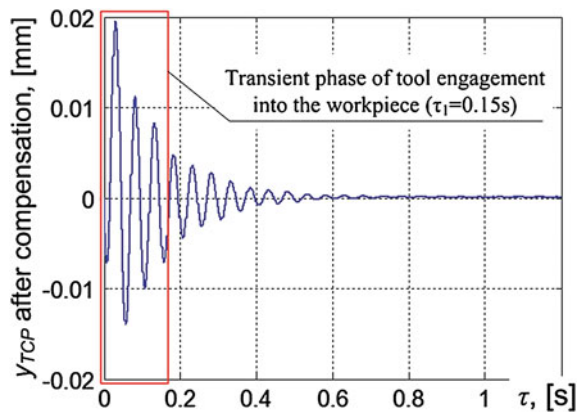


Table 13.2 Milling trajectory accuracy before and after compliance error compensation

Performance measure	Original trajectory	Modified trajectory
Low frequency (Hz)	6.70	6.70
Static deviation y_s (mm)	58.1e-3	0.14e-3
Max deviation y_{MAX} (mm)	63.2e-3	4.70e-3

is analyzed here (Fig. 13.15). Comparison results presented in Figs. 13.13, 13.15 are summarized in Table 13.2. So after applying error compensation technique the static deviation in y direction has been reduced from 0.058 to 0.00014 mm (99.8%). Maximum defilation in the machining profile has been reduced from 0.063 to 0.0047 mm (92.6%). Low frequency remained the same for both cases.

Hence, obtained results show that the developed compliance error compensation allows us significantly increase the accuracy of the robotic-based machining.

13.7 Conclusions

In robotic-based machining, an interaction between the workpiece and technological tool causes essential deflections that significantly decrease the manufacturing accuracy. Relevant compliance errors highly depend on the manipulator configuration and essentially differ throughout the workspace. Their influence is especially important for heavy serial robots. To overcome this difficulty this chapter presents a new technique for compensation of the compliance errors caused by technological process. In contrast to previous works, this technique is based on the non-linear stiffness model and the reduced elasto-dynamic model of the robotic based milling process.

The advantages and practical significance of the proposed approach are illustrated by milling with of KUKA KR270. It is shown that after error compensation technique

significantly increase the accuracy of milling. In future the proposed technique will be integrated in a software toolbox.

Acknowledgments The authors would like to acknowledge the financial support of the ANR, France (Project ANR-2010-SEGI-003-02-COROUSSO), FEDER ROBOTEX, France and the Region “Pays de la Loire”, France.

References

1. Dépincé, P., Hascoët, J.-Y.: Active integration of tool deflection effects in end milling. Part 2. Compensation of tool deflection. *Int. J. Mach. Tools Manuf.* **46**, 945–956 (2006)
2. Altintas, Y.: *Manufacturing Automation, Metal Cutting Mechanics, Machine Tool Vibrations and CNC Design*. Cambridge University Press, New York (2000)
3. Eastwood, S.J., Webb, P.: A gravitational deflection compensation strategy for HPKMs. *Robot. Comput. Integr. Manuf.* **26**, 694–702 (2010)
4. Alici, G., Shirinzadeh, B.: Enhanced stiffness modeling, identification and characterization for robot manipulators. In: *Proceedings of IEEE Transactions on Robotics* vol. 21, pp. 554–564 (2005)
5. Chen, S., Kao, I.: Conservative congruence transformation for joint and cartesian stiffness matrices of robotic hands and fingers. *Int. J. Robot. Res.* **19**(9), 835–847 (2000)
6. Kövecses, J., Angeles, J.: The stiffness matrix in elastically articulated rigid-body systems. *Multibody Sys. Dyn.* **18**(2), 169–184 (2007)
7. Pashkevich, A., Klimchik, A., Chablat, D.: Enhanced stiffness modeling of manipulators with passive joints. *Mech. Mach. Theory* **46**(5), 662–679 (2011)
8. Tyapin, I., Hovland, G.: Kinematic and elastostatic design optimization of the 3-DOF Gantry-Tau parallel kinematic manipulator. *Model. Ident. Control* **30**(2), 39–56 (2009)
9. Su, H.-J., McCarthy, J.M.: A polynomial homotopy formulation of the inverse static analyses of planar compliant mechanisms. *J. Mech. Des.* **128**(4), 776–786 (2006)
10. Briot, S., Pashkevich, A., Chablat, D.: Reduced elastodynamic modelling of parallel robots for the computation of their natural frequencies. In: *13th World Congress in Mechanism and Machine Science*, pp. 19–25. Guanajuato, Mexico (2011)
11. Ritou, M., Garnier, S., Furet, B., Hascoët, J.Y.: A new versatile in-process monitoring system for milling. *Int. J. Mach. Tools Manuf.* **46**(15), 2026–2035 (2006)
12. Brissaud, D., Goukov, A., Paris, H., Tichkiewitch, S.: The fractional model for the determination of the cutting forces. *Asian Int. J. Sci. Technol. Prod. Manuf.* **1**, 17–25 (2008)
13. Merchant, M.E.: Mechanics of metal cutting process. I-Orthogonal cutting and type 2 chip. *J. Appl. Phys.* **16**(5), 267–275 (1945)
14. Laporte, S., Knevez, J.-Y., Cahuc, O., Darnis, P.: Phenomenological model for drilling operation. *Int. J. Adv. Manuf. Technol.* **40**, 1–11 (2009)
15. Tlustý, J., Ismail, F.: Basic non-linearity in machining chatter. *Ann. CIRP* **30**(1), 299–304 (1981)
16. Gong, C., Yuan, J., Ni, J.: Nongeometric error identification and compensation for robotic system by inverse calibration. *Int. J. Mach. Tools Manuf.* **40**(14), 2119–2137 (2000)
17. Dumas, C., Caro, S., Garnier, S., Furet, B.: Joint stiffness identification of six-revolute industrial serial robots. *Robot. Comput. Integr. Manuf.* **27**(4), 881–888 (2011)

Chapter 14

A Modified LGMD Based Neural Network for Automatic Collision Detection

Ana Carolina Silva, Jorge Silva and Cristina Peixoto dos Santos

Abstract Robotic collision detection is a complex task that requires both real time data acquisition and important features extraction from a captured image. In order to accomplish this task, the algorithms used need to be fast to process the captured data and perform real time decisions. Real-time collision detection in dynamic scenarios is a hard task if the algorithms used are based on conventional techniques of computer vision, since these are computationally complex and, consequently, time-consuming, specially if we consider small robotic devices with limited computational resources. On the other hand, neurorobotic models may provide a foundation for the development of more effective and autonomous robots, based on an improved understanding at the biological basis of adaptive behavior. Particularly, our approach must be inspired in simple neural systems, which only requires a small amount of neural hardware to perform complex behaviours and, consequently, becomes easier to understand all the mechanism behind these behaviours. By this reason, flying insects are particularly attractive as sources of inspiration due to the complexity and efficiency of the behaviours allied with the simplicity of a reduced neural system. The Lobula Giant Movement Detector (LGMD) is a wide-field visual neuron located in the Locust optic lobe. It responds selectively to looming objects and can trigger avoidance reactions when a rapidly approaching object is detected. Based on the relatively simple encoding strategy of the LGMD neuron, different bio-inspired neural networks for collision avoidance were developed. In the work presented in this chapter, we propose a new LGMD model based on two previous models, in order to improve over them by incorporating other features. To accomplish this goal, we proceed as follows: (1) we critically analyse different LGMD models proposed

A. C. Silva (✉) · J. Silva · C. P. dos Santos
Industrial Electronic Department, University of Minho, Guimarães, Portugal
e-mail: ana.silva@dei.uminho.pt

J. Silva
e-mail: jbruno@dei.uminho.pt

C. P. dos Santos
e-mail: cristina@dei.uminho.pt

in literature; (2) we highlight the convergence or divergence in the results obtained with each of the models; (3) we merge the advantages/disadvantages of each model into a new one. In order to assess the real-time properties of the proposed model, it was applied to a real robot. The obtained results have shown the high capability and robustness of the LGMD model to prevent collisions in complex visual scenarios.

Keywords Bio-inspired model · Lobula Giant Movement Detector neuron · Artificial neural networks · Collision avoidance

14.1 Introduction

Many animals extract salient information from complex, dynamic visual scenes to drive behaviours necessary for survival. Insects are particularly challenging for robotic systems: they achieve their performance with a nervous system that has less than a million neurons and weighs only about 0.1 mg. By this reason, some of these insects provide ideal biological models that can be emulated in artificial systems. These models have the potential to reproduce complex behaviours with low computational overhead by using visual information to detect imminent collisions caused either by a rapidly approaching object or self-motion towards an obstacle.

In locusts, the Lobula Giant Movement Detector (LGMD) is a bilaterally paired motion sensitive neuron that integrates inputs from the visual system, responding robustly to images of objects approaching on a collision course [1–4]. This neuron is responsible for triggering escape and collision avoidance behaviours in locusts.

The first physiological and anatomical LGMD neuron model was developed by Bramwell in [5]. The model continued to evolve [6–9] and it was used in mobile robots and deployed in automobiles for collision detection. These connectionist models have shown that the integration of on and off channels and feed-forward inhibition can account for aspects of the LGMD neuron looming sensitivity and selectivity when stimulated with approaching, translating and receding objects. However, further work is needed to develop more robust models that can account for complex aspects of visual motion [10].

In this chapter, we are interested in integrating two previous LGMD models in order to take the advantage of noise immunity proposed in [7] and direction sensitivity proposed in [9]. In a previous study, we implemented the models from [7, 9] and submitted them to relevant simulated visual data sets. This step enabled us to understand some of the literature models limitations in relation to obstacle detection and avoidance. With this knowledge, we herein propose a new model to cope with the limitations showed by the models implemented. The proposed model is validated over a set of different visual scenarios. In order to the LGMD network be used as a robust collision detector for real robotic applications, and based on [7], a new mechanism to enhance features of colliding objects was proposed. The model from [7] favours grouped excitation, which normally indicates the presence of an obstacle, and ignores isolated excitation, which can be the result of noise present in the captured

image. This model has the capability to filter out the isolated excitations through an excitation gathering mechanism, allowing that only parts of the captured image with bigger excitatory spatial areas can contribute to the excitation of the LGMD cell. Besides this extraordinary capability of noise reduction, when computationally implemented, the neural network based on [7] generated false collision alarms when stimulated with receding objects. The LGMD model proposed in [9] is able to detect the direction of movement in depth. However, the latter is not immune to the presence of noise levels in the captured image, which can lead it to produce false collision alerts in the presence of noise. Based on [9], we have modified the LGMD model, so that it can distinguish approaching from receding objects.

Merging the advantages of each model [7, 9], the proposed LGMD model is more robust in collision detection. The proposed model is tested on simulated and real video recording environments. The obtained results show that it works very efficiently in both scenarios. The real performance of the proposed model is judged by the evaluation of a real robot moving around in a real environment and avoiding real obstacles (of different shapes, sizes and colours) while processing captured images (containing real noise, blur, reflections, etc).

The new proposed method increases the precision of obstacle detection, in a way that this model is robust to the presence/absence of high noise levels in the captured image, as well as being able to detect the movement direction of the visual stimulus. Besides that, when tested in a real environment, the results were very satisfactory. For a better understanding of the work developed, the chapter was organized in the following way: in Sect. 14.2, we make a detailed description of the proposed LGMD neural network model. In Sect. 14.3 are presented some experimental results on simulated and recorded video data. Additionally, it is also presented the experiments carried out with a robot DRK8000 to test the stability of this model in relation to collision detection in real scenarios. Finally, in Sect. 14.4 we discuss the conclusions of the work here described.

14.2 The Proposed LGMD Neural Network

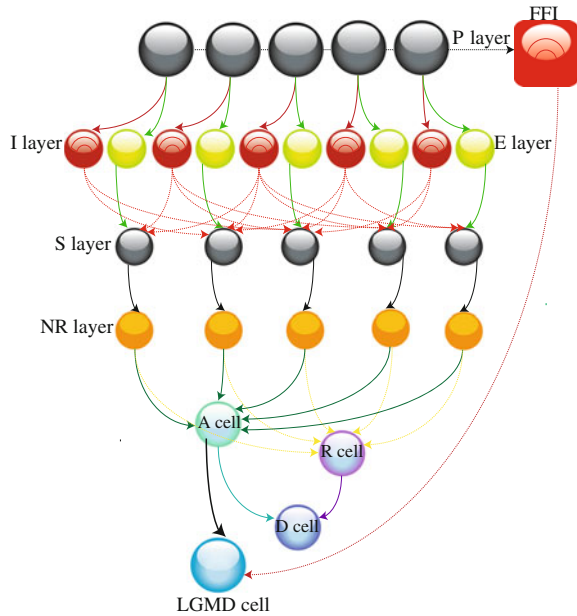
The biological inspired neural network here proposed is based on previous models described on [6–9]. The modified neural network is shown on Fig. 14.1.

The LGMD neural network here proposed is composed by five groups of cells: photoreceptor cells (P layer), excitatory cells (E layer), inhibitory cells (I layer), summing cells (S layer) and noise reduction cells (NR layer). Besides that, it is composed by five single cells: the direction sensitive system, composed by the approaching cell (A cell), the receding cell (R cell) and the direction cell (D cell), the feed-forward inhibition cell (FFI cell) and the LGMD cell.

A grayscale image of the camera current field of view, represented has a matrix of values (from 0 to 255), is the input to a matrix of photoreceptor units (P layer).

This layer calculates the absolute difference between the luminance of the current and of the previous input images, mathematical represented by the following

Fig. 14.1 Schematic illustration of the proposed LGMD model



equation:

$$P_f(x, y) = |L_f(x, y) - L_{f-1}(x, y)|, \tag{14.1}$$

where $P_f(x, y)$ is the output relative to the cell in the (x, y) position at frame f , $L_f(x, y)$ and $L_{f-1}(x, y)$ are the captured luminance at position (x, y) for frames f and $f - 1$, respectively. The output of the P layer is the input of two different layers: the excitatory (E) and the inhibitory (I) layer. To the excitatory cells of the E layer, the excitation that comes from the P layer is passed directly to the retinotopic counterpart at the S layer. And the inhibition layer (or I layer) receives the output of the P layer and applies a blur effect on it, using:

$$I_f(x, y) = \sum_{i=-1}^1 \sum_{j=-1}^1 P_{f-1}(x+i, y+j) \cdot W_I(i, j), \quad i, j \neq 0, \tag{14.2}$$

where $I_f(x, y)$ is the inhibition relative to the cell in the (x, y) position at frame f , $W_I(i, j)$, an empirically set kernel, represents the local inhibition weight. The inhibition from the (x, y) cell only spreads to the nearest neighbors and does not inhibits itself. This process is strongly supported by the biological nervous systems. In biology, an excited neuron does not inhibits itself, it inhibits the neighboring neurons, with a temporal delay associated to the inhibitory synapses (and this is the reason why we use the P cell excitement corresponding to the previous time-step, $f - 1$). Relative to the definition of the values holding by this kernel, the inhibition

value of a particular cell is given by the distance at which a neighboring cell is located. The use of such kernel is also based on biological systems, since distant neurons inhibit a particular neuron with less intensity than those that are closest to a neuron, due to the decrement of the neuronal signal with increasing distance. Finally, the excitatory flux from the E cells and the inhibition that comes from the I cells are summed by the S cells (summing cells), using the following equation:

$$S_f(x, y) = E_f(x, y) - w_i \cdot I_f(x, y), \quad E_f(x, y) = P_f(x, y), \quad (14.3)$$

where w_i (a scalar) represents the inhibition strength. Based on [7], a new mechanism for the LGMD neural network was added to filter background noise. This mechanism, implemented in the NR layer, takes clusters of excitation in the S units to calculate the input to the LGMD membrane potential. These clusters provide higher individual inputs than the ones of isolated S units. The excitation that comes from the S layer is then multiplied by a passing coefficient Ce_f , whose value depends on the surrounding neighbours of each pixel, calculated as follows:

$$Ce_f(x, y) = \frac{1}{9} \sum_{i=-1}^1 \sum_{j=-1}^1 S_f(x+i, y+j) \quad (14.4)$$

The final excitation level of each cell in the NR (Noise-Reduction) layer, at frame f (NR_f), is given by:

$$NR_f(x, y) = |S_f(x, y) \cdot Ce_f(x, y) \cdot w^{-1}| \quad (14.5)$$

$$w = \max(|Ce_f|) C_w^{-1} + \Delta c \quad (14.6)$$

C_w is set to 4, Δc is a small number (0.01) to prevent w from being zero, and $\max(|Ce_f|)$ is the largest element in matrix $|Ce_f|$. Within the NR layer, a threshold filters the decayed excitations (isolated excitations), as:

$$\tilde{NR}_f(x, y) = \begin{cases} NR_f(x, y), & \text{if } NR_f(x, y) \cdot C_{de} \geq T_{de} \\ 0, & \text{if } NR_f(x, y) \cdot C_{de} < T_{de} \end{cases}, \quad (14.7)$$

where $C_{de} \in [0, 1]$ is the decay coefficient and T_{de} is the decay threshold (set to 20). The decay threshold here used was experimentally determined. The NR layer is able to filter out the background detail that may cause excitation. Hence, only the main object in the captured scene will cause excitation. The $LGMD$ potential membrane K_f , at frame f , is summed after the NR layer,

$$LGMD_f = K_f = \sum_{x=1}^n \sum_{y=1}^m (\tilde{NR}_f(x, y)), \quad (14.8)$$

where n is the number of rows and m is the number of columns of the matrix representing the captured image. The A (Approaching) and R (Receding) cells (adapted from [9]) are two grouping cells for depth movement direction recognition. The A cell holds the mean of three samples of the $LGMD$ cell.

The R cell shares the same structure as the A cell but with a temporal difference, having one frame delay from A . According to the theory described above, it can be concluded that if the object is approaching $A_f > R_f$ and if the object is receding, $R_f > A_f$. The D cell or Direction cell ($\in \{-1, 0, 1\}$ in case of receding, no movement and approaching object, respectively) is used to calculate the direction of movement. This cell exploits the movement direction in depth. It is based on the fact that a looming object (approaching) gets larger whereas a receding object gets smaller. In a way to distinguish the movement direction detected by the D cell, a threshold mechanism was added, $T_D(0.05 \times n \times m)$, which was experimentally determined.

$$D_f = \begin{cases} 1, & \text{if } |A_f| - |R_f| \geq T_D \\ 0, & \text{if } T_D < |A_f| - |R_f| < T_D \\ -1, & \text{if } |A_f| - |R_f| \leq T_D \end{cases} \quad (14.9)$$

The $LGMD$ membrane potential K_f is then transformed to a spiking output $k_f \in [0.5, 1]$ using a sigmoid transformation,

$$k_f = (1 + e^{-K_f \cdot ncell^{-1}})^{-1}, \quad (14.10)$$

where $ncell$ is the total number of cells in the NR layer. The collision alarm is decided by the spiking of the $LGMD$ cell.

A spiking mechanism was implemented using an adaptable threshold. This threshold starts with a value experimentally determined, T_s (0.88) and it is updated at each frame, through the following process,

$$T_s = \begin{cases} T_s + \Delta t, & \text{if } s_{av} > \Pi \text{ and } (T_s + \Delta t) \in [T_l, T_u] \\ T_s - \Delta t, & \text{if } s_{av} < \Pi \text{ and } (T_s - \Delta t) \in [T_l, T_u], \\ T_s, & \text{others} \end{cases} \quad (14.11)$$

where $[T_l, T_u]$ defines the lower and upper limits for adaptation (T_l is 0.80 and T_u is 0.90), $\Delta t = 0.01$ is the increasing step, $\Pi = 0.72$ is a threshold that limits the averaged spiking output s_{av} , between frame $f - 5$ and frame $f - 2$,

$$s_{av} = 0.25 \sum_{i=2}^5 s_{f-i}. \quad (14.12)$$

If the sigmoid membrane potential k_f exceeds the threshold T_s a spike is produced, as follows:

$$s_f = \begin{cases} 1, & \text{if } k_f \geq T_s \\ 0, & \text{others} \end{cases} \quad (14.13)$$

Finally, a collision is detected when there are n_{sp} spikes in n_{ts} time steps ($n_{sp} \leq n_{ts}$), where n_{sp} is 4 and n_{ts} is 5 (values experimentally determined).

$$C_f = \begin{cases} 1, & \text{if } \sum_{f-n_{ts}}^f s_f \geq n_{sp} \\ 0, & \text{others} \end{cases} \quad (14.14)$$

The escape behavior is initialized when a collision is detected. Besides that, the spikes can be suppressed by the *FFI* cell when whole field movement occurs. If it is not suppressed during the tuning of the robot, for example, the network may produce spikes and even false collision alerts due to sudden changes in the visual scenario.

The *FFI* cell is a cell which is very similar to the *LGMD* cell but receives the output from the *P* layer (and not from the *S* layer), as follows:

$$FFI_f = \frac{\sum_{x=1}^m \sum_{y=1}^n |P_{f-1}(x, y)|}{ncell}, \quad (14.15)$$

where P_{f-1} is the output of the *P* layer at frame $f - 1$. If FFI_f exceeds a threshold T_{FFI} (experimentally set to 25), the spikes produced by the *LGMD* cell are automatically inhibited.

As described in this section, the proposed neural network for the *LGMD* neuron only involves low level image processing. So, the proposed neural network model is able to work in real time and, besides that, is independent of object classification.

14.3 Experimental Results on the Proposed Model

In a way to test the efficiency of the *LGMD* neural network here proposed, two experimental scenarios were used. The first experiment was made on a simulated data set and, after that, a recorded video was used to prove the capability of the *LGMD* neural network here proposed to work in a real environment. In the second experiment, the *LGMD* neural network was implemented in a real robot, DRK8000, located within a real arena. All the model parameters were kept the same during all the experiments.

14.3.1 Simulated Environment

We developed a simulation environment in Matlab, which enables us to assess the effectiveness of the proposed *LGMD* neural network. Objects were simulated accord-

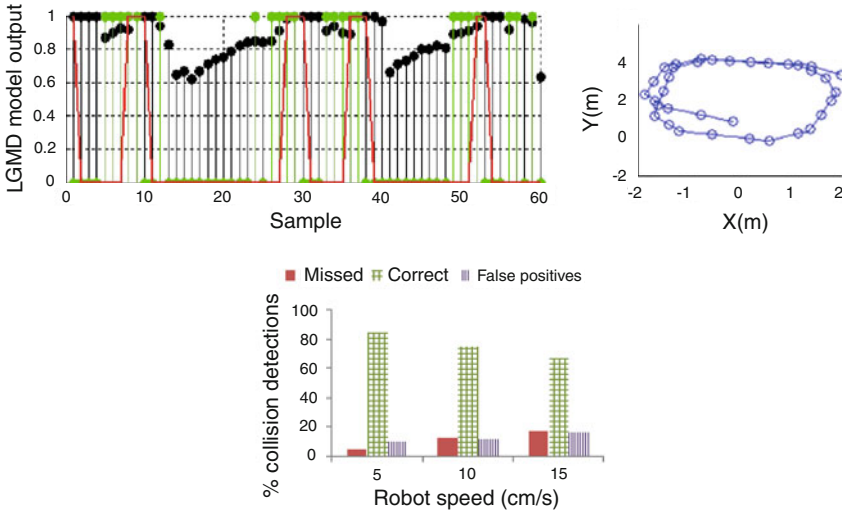


Fig. 14.2 Artificial visual stimuli, developed in Matlab

ing to their movement and the corresponding data was acquired by a simulated camera and processed by the LGMD neural network. Image sequences were generated by a simulated camera with a field of view of 60° , both in x and y axis, using a sample frequency of 100 Hz.

The simulated environment enabled us to adjust several parameters, such as: image matrix dimensions, the camera rate of acquisition, the image noise level, the object shape, the object texture, as well as other parameters.

The computer used was a Laptop (Toshiba Portegé R830-10R) with 4 GHz CPUs and Windows 7 operating system. Relative to the parameters used by the LGMD neural network, they were determined before the experiments.

14.3.2 LGMD Model Validation

Previous to the stimulation of the LGMD model here proposed, several experiments have been made in order to verify and analyse how the image of a black squared object grows when it is approaching a simulated camera. For that, we used synthesized black (0) and white (255) images (see Fig. 14.2), with 100 (horizontal) by 100 (vertical) pixels of resolution. The object being observed was a square black filled rectangle, whose properties as acquisition frequency, velocity, trajectory, shape, texture, noise level or object size could be changed. The obtained results enabled us to conclude that the image growing can be approximated by an exponential curve, whose slope depends on several factors, including the camera acquisition frequency

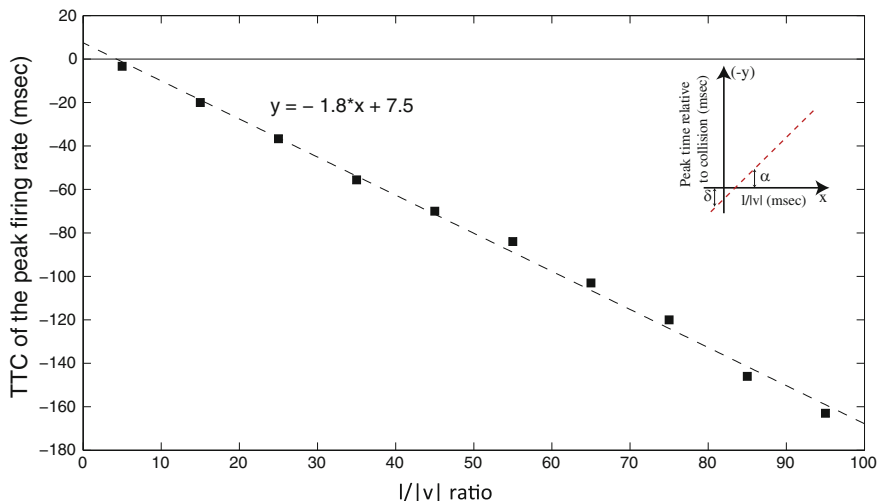


Fig. 14.3 Dependence of peak firing time relative to collision on $l/|v|$ ratio obtained with the LGMD model here proposed

and the object velocity, among other characteristics. However, the curve that approximates this growing is always an exponential curve, whose slope depends on these aforementioned factors.

As a second step, and in the context of this study, we made an exhaustive analysis of the response of our LGMD model to a set of standard LGMD stimulation protocols, which allowed us to validate our model with respect to the biological system [11, 12]. Firstly, we evaluated the proposed LGMD model, by using a looming stimulus consisting of a solid square with 10 repetitions to each size/velocity = $l/|v|$ pair (where l stands for the half length of the square object and v for its linear velocity). With these experiments, we wanted to prove that our model respects the properties verified by Gabbiani et al. [3, 11] as well as by Badia [13]. These properties, founded in the locust visual system, include a linear relation between the time of the peak firing rate of the LGMD neuron and the ratio that correlated the stimulus object size (l) and the stimulus linear velocity (v) [12]. As a first step, we analysed the LGMD model here proposed using a looming stimulus in the form of a black square. We repeated this procedure to ten different $l/|v|$ ratios, from 5 to 95 ms, in steps of 10 ms.

Through the obtained results it was observed that the fit of the TTC (time-to-collision) of the peak firing rate, obtained through the LGMD neural network, versus the $l/|v|$ ratios, is consistent with the biological results (Fig. 14.3), showing a correlation coefficient (r) superior to 0.99. By performing the analysis developed by Gabbiani [3, 12], based on the regression line represented on Fig. 14.3 ($y = -1.8x + 7.5$ with a correlation coefficient equal to 0.9981) and comparing it to the linear regression, represented on Fig. 14.3 (top right) we can conclude that α (representing the slope) takes the value -1.8 and the δ (intercept value) takes the value 7.5 ms. Using these values, and taking in consideration the Gabbiani formu-

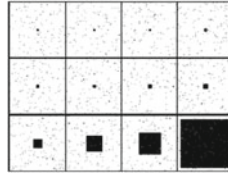


Fig. 14.4 Selected frames from the simulated image sequence. The *square* object changes its size from *small* (10 by 10 cm, $l = 5$ cm) to *big*, and moves at 100 cm/s ($v = 100$ cm/s). The relation $l/|v|$ is 50 ms. The noise level in all the image sequence is, approximately, 500 pixels. The frame rate was 100Hz

lation, which says that angular size (named $\theta_{threshold}$) subtended by the approaching object δ milliseconds before the LGMD output reach the peaks, is given by: $\theta_{threshold} = 2 \cdot \text{atan} \cdot 1/\alpha$, we conclude that the peak firing rate always occurred 7.5 ms after the object had reached a full angular size of 60° on the camera, for all $l/|v|$ values.

If the angular threshold is used by the LGMD model to trigger escape responses, it should be encoded independently of the looming stimuli particular properties. In order to verify this hypothesis, ie, the invariant properties of the LGMD model response to the shape, texture and approaching angle of the visual stimulus, a series of experiments were done. For that, four different stimulus were developed in Matlab: the first is the one previously described, a single black square, with a white background, approaching at different $l/|v|$ ratios. The second stimulus developed is a black circle, with a white background, approaching also at different $l/|v|$ ratios. The third visual stimulus is a square with a checkerboard texture. Finally, the last visual stimulus is a simple square deviated 50° relatively to the center of the camera that generates the visual stimuli (see Fig. 14.2).

According to the obtained results, we verify that the linear relation between the TTC of peak firing rate and the $l/|v|$ ratio was not affected by the shape, texture or approaching angle of the visual stimuli ($r \approx 0.99$). Although, we also observed that the activity of the LGMD model was significantly reduced for the case of a misalignment of 75° or more, due to the loss of stimulation by the looming stimulus (as the object approaches, part of the visual stimuli remains outside the screen).

According to the obtained results previously described, we conclude that the intrinsic linear dependence between the peak firing time and the $l/|v|$ ratio remains preserved by the LGMD model here proposed.

14.3.3 Results on Simulated Data Set and on Real Recorded Data

After the model validation, we fed the LGMD neural networks proposed by [7, 9] and the one proposed by us, with simulated image sequences (a representation can be observed on Fig. 14.4).

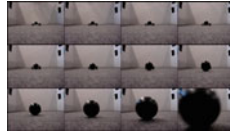


Fig. 14.5 Selected frames from the recorded image sequence used in the experiment. The recorded video is composed by 44 frames, showing a black approaching ball

Table 14.1 Distances at which collision detection alarms were generated by the LGMD model 1 and LGMD model 2, in five different situations tested

	LGMD model 1 (cm)	LGMD model 2 (cm)
Stimulus 1	26	14
Stimulus 2	35	–
Stimulus 3	26	20
Stimulus 4	35	11
Real video	24	14

In this point, we used four different simulated visual stimuli: Stimulus 1: composed by a black approaching square, over a white background, with l/v equal to 50 ms, acquired with a frame rate of 100 Hz, without noise added to the image sequence. Stimulus 2: composed by a black receding square, over a white background, with l/v equal to 50 ms, acquired with a frame rate of 100 Hz, without noise added to the image sequence. Stimulus 3: composed by a black approaching square, over a white background, with l/v equal to 50 ms, acquired with a frame rate of 100 Hz, with 500 pixels of noise added to the image sequence. Stimulus 4: composed by a black receding square, over a white background, with l/v equal to 50 ms, acquired with a frame rate of 100 Hz, with 500 pixels of noise added to the image sequence.

In addition to these four simulated visual stimuli, and in order to test the LGMD models in a real environment, we recorded a real video sequence, using a Sony Cyber shot digital camera 7.2 megapixels to obtain the video clip. The resolution of the video images was 640 by 480 pixels, with an acquisition frequency of 30 frames/s. In Fig. 14.5 it is represented some selected frames captured by the camera, showing a real approaching black ball.

After the computational implementation of the LGMD models proposed in [7] and [9], and after subject those to all the stimuli previously described, we verify that the collisions were detected, by the different LGMD models, at different time instants and, consequently, at different distances of the object (simulated or real) relatively to the camera. For a better understanding and organization of the results, we decided to call “LGMD model 1” to the model proposed by [7] and “LGMD model 2” to the model proposed by [9]. The results obtained are resumed in Table 14.1.

As we can observe on Table 14.1, in the approaching situations (stimulus 1, 3 and real video), the LGMD model 1 detected a collision when the object was located at, approximately, 24–26 cm relatively to the camera. This model showed its immunity

to the noise presence since it detected a collision exactly at the same distance when stimulated with stimulus 1 (absence of noise) and 3 (presence of high noise levels). However, if we observe the obtained results for the LGMD model 1 when stimulated with receding objects (stimulus 2 and 4) it detected a false collision when the object was located at 35 cm relatively to the camera, in both situations tested. Through these last results one can conclude that the LGMD model 1 is not able to distinguish between approaching and receding objects, generating false collision alerts in the presence of receding objects. But we can also conclude that this model has high immunity to the noise presence in the captured images.

Relatively to the LGMD model 2 and observing Table 14.1, for the stimulus 1 and 3, this model did not detect collisions for the same distance. When stimulated with stimulus 1, it detected a collision when the object was located at 14 cm relatively to the camera and when stimulated with stimulus 3, a collision was detected sooner, when the object was at 20 cm relatively to the camera. This happened due to the fact that the LGMD model 2 is not immune to the noise presence and the noise pixels, which were not eliminated by this model, composed an extra excitation to the LGMD neural network.

In the presence of a receding object, the LGMD model 2 was able to not produce false collision alerts when stimulated with stimulus 2. However, when we feed the LGMD model 2 with the stimulus 4, it detected a false collision when the object was located at 11 cm relatively to the camera. This happened also due to the non-immunity of the LGMD model 2 to the noise presence, which works as an extra excitation, leading to the generation of false collision alerts.

After this analysis, relative to the behaviour of the LGMD model 1 and LGMD model 2 in different situations, we could extract some particular characteristics of both models. These results led us to produce a mixed LGMD model, combining the advantages of the LGMD model 1 and LGMD model 2. Thus, the LGMD model here proposed provides noise immunity, as well as a directionally sensitive system.

Figure 14.6 shows the output from the LGMD model here proposed. In this figure, at each time step we can observe the result of different mathematical processing (described on Sect. 14.2), corresponding to the layers of the proposed model, executed sequentially, necessary to detect, with the maximum precision, an imminent collision.

The analysis of these results showed, on Fig. 14.6, that the LGMD neural network detected a collision at time -0.19 s, i.e., when the object was located at 19 cm relatively to the camera. In relation to the receding object, represented on Fig. 14.7, no collisions were detected, as expected.

The results previously described showed the efficacy of the LGMD neural network proposed by us. On Figs. 14.6 and 14.7, it is shown the LGMD model immunity to high noise levels, as well as the capability of this model in distinguish the direction of movement between successive frames. Then, to test the capability of the proposed LGMD model in a more realistic environment, we subjected it to the real video sequence, represented on Fig. 14.5. In this situation, the model produced a collision alert when the object was located at 28 cm relatively to the camera.

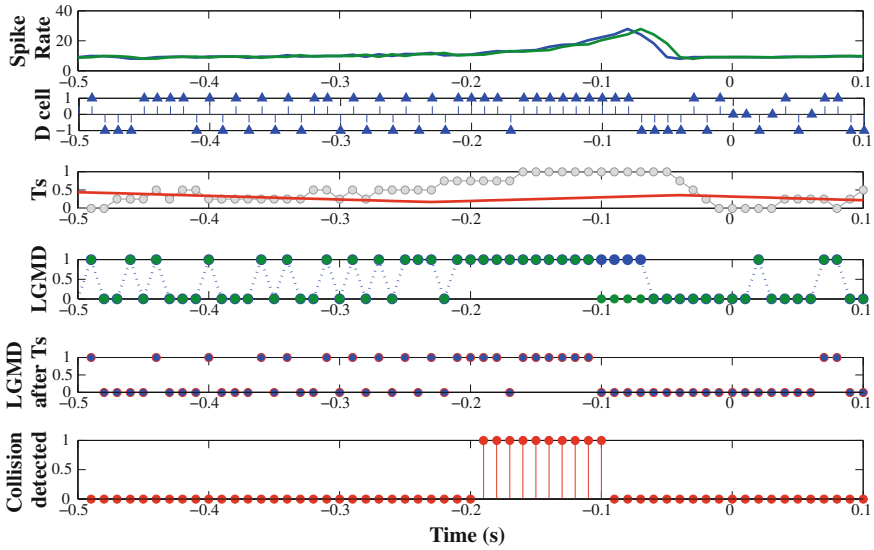


Fig. 14.6 LGMD model response to an approaching object which $l/|v|$ set at 50ms. Spike Rate: *blue graph*: is obtained by the ratio of the A cell value and the total number of cells in the NR layer. *Green graph*: is obtained by the ratio of the R cell value and the total number of cells in the NR layer. D cell: output of the direction cell: 1: approaching, 0: no significant movement, -1: receding. Ts: adaptative threshold represented by the *red line*; the *gray points* represent the sav output. LGMD: *Blue graph*: output of the LGMD cell (mathematically represented by the k_f value). *Green points*: output of the LGMD cell after the Feed-forward inhibition. LGMD after Ts: represents the output of the LGMD cell after the application of the threshold Ts and being in account the output of the D cell. Collision detected: the output of this graph is one when it is detected four successive spikes in five successive time-steps. In all these graphs, the zero value corresponds to the time of collision

14.3.4 Results on a Real Robot

In order to assess the capability of the LGMD model here proposed in a real environment, we used a DRK8000 mobile robot, with a 8-bit CIF (352 by 288 pixels) colour CMOS camera, working at 10Hz, having a field-of-view of 70° , approximately. The robot was located within an arena, surrounded by four walls with attached objects with different colours, shapes, textures and sizes. The arena has 16m^2 . We used the dead reckoning process in order to predict the position of the robot at each time instant.

As Fig. 14.8 shows, the simulation system used comprises four modules: the LGMD model, the robot control, the tracking and the graphical user interface. The experiment ran in real-world time, with 10 time steps per second. The LGMD model module was composed by the different layers observed on Fig. 14.1, and the final output of this model comprises two different states: “collision detected” or “non-collision detected”.

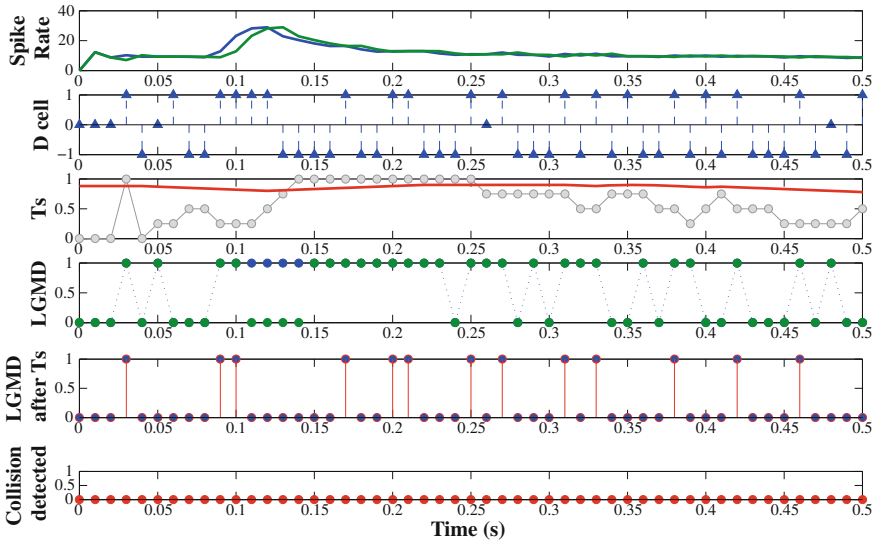
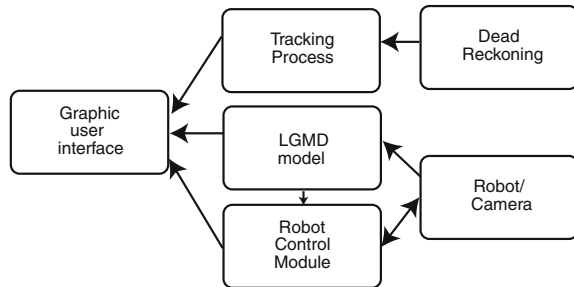


Fig. 14.7 LGMD model response to a receding object which $l/|v|$ was equal to 50 ms. The legend of this figure is similar to the one described on the Fig. 14.6

Fig. 14.8 Integrated simulation processes used in the real experiment



The robot control module consists in the reactive control structure, capable of controlling the robot, using only the output of the LGMD model module. The behaviours comprised by this module, can be divided in two: (1) basic exploratory activity and (2) collision avoidance of obstacles, triggered by the response of the LGMD module. If the robot detects an imminent collision, it stops, rotates and, then, continues the movement in a straight line. The turning speed is $1/3$ of the robot speed for the left wheel and $-1/3$ of the robot speed to the right wheel. The robot was set to rotate during 1 s. Finally, in relation to the tracking process, we used dead reckoning in order to determine the position of the robot at each time step and, then, use this information to infer about the distance at which the robot deviates of a potential collision/obstacle.

In the experiment, three long robot movement periods (120 s, speed at 5, 10 and 15 cm/s) were conducted to test and show the mechanism of the collision detector

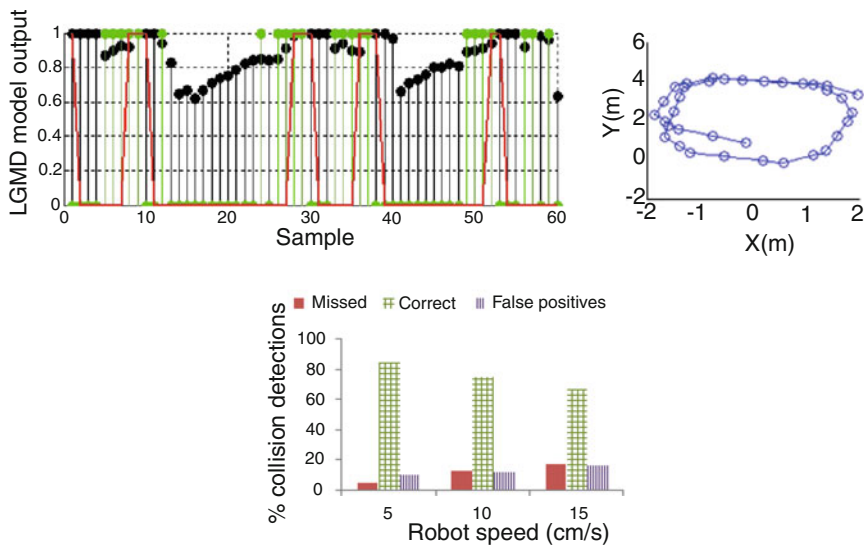


Fig. 14.9 *Top graph:* LGMD model output, running at real time, for different LGMD layers, during the experiment with the DRK8000 robot, for a robot speed of 5 cm/s. *Middle graph:* Dead reckoning of the robot during the initial time steps of the experiment, for a robot speed of 5 cm/s. Categorization of the collision detections as missed, correct and false positives, for three different robot velocities tested: 5, 10 and 15 cm/s

in a real environment. After the experiment, and through the analysis of the dead reckoning relative to the robot movement during all the running time, we could extract, as well as characterize, the collision detections. Collision detections between 20 and 100cm away from the wall were classified as correct, those detected closer than 20cm from the wall were classified as missed, and collisions detected at a distance over 100cm as false positives (see Fig. 14.9).

As represented on Fig. 14.9, as the velocity of the robot increases, the percentage of collision detections classified as correct decreases, as well as the percentage of missed and false positives detections increases. The increase of missed collisions to higher speeds was due to the decrease of the $l/|v|$ ratio (for the same objects within the arena, l variable keeps the same in different experiments, but as the velocity (v) increases the ratio decreases), leading the LGMD firing rate reach the peak nearer to the time predicted to collision (as we can observe on Fig. 14.3). Due to this reason, more collisions are missed for higher velocities. The increase in the number of false positives to higher velocities is based on the fact that, at higher velocities, the difference between successive frames is higher, leading to the production of high excitation levels and, consequently, a higher number of collision detection alarms.

Although the difference verified in relation to correct collision detections between different velocities, the results obtained are very satisfactory, as the number of correct detections are always higher than the sum of missed and false positive detections.

14.4 Conclusions

In this chapter, we propose a modified LGMD model based on the identified LGMD neuron of the locust brain. The model proved to be a robust collision detector for autonomous robots. This model has a mechanism that favours grouped excitation, as well as two cells with a particular behaviour that provide additional information on the depth direction of movement. For applications as collision detectors in robotics, the model proposed is able to remove the noise captured by the camera, as well as enhance its ability to recognize the direction of the object movement and, by this way, remove the false collision alarms produced by the previous models when a nearby object is moving away. Experiments with a DRK8000 robot showed that with these two new procedures, the robot was able to travel autonomously in real time and within a real arena. The results illustrate the benefits of the LGMD based neural network here proposed, and, in the future, we will continue to use and enhance this approach, using, for that, a combination of physiological and anatomical studies of the locust visual system, in order to improve our understanding about the relation between the LGMD neuron output and the locust muscles related to the avoidance manoeuvres.

Acknowledgments Ana Silva is supported by Ph.D. Grant SFRH/BD/70396/2010. This work is funded by FEDER Funds through the Operational Programme Competitiveness Factors—COMPETE and National Funds through FCT—Foundation for Science and Technology under the Project: FCOMP-01-FEDER-0124-022674.

References

1. Gray, J.R., Lee, J.K., Robertson, R.M.: Activity of descending contralateral movement detector neurons and collision avoidance behaviour in response to head-on visual stimuli in locusts. *J. Comp. Physiol. A* **187**(2), 115–129 (2001)
2. Rind, F.C.: Non-directional, movement sensitive neurones of the locust optic lobe. *J. Comp. Physiol. A* **161**(3), 477–494 (1987)
3. Gabbiani, F., Krapp, H., Laurent, G.: Computation of object approach by a wide-field motion-sensitive neuron. *J. Neurosci.* **19**, 1122–1141 (1999)
4. Gray, J.R., Blinow, E., Robertson, R.: A pair motion-sensitive neurons in the locust encode approaches of a looming object. *J. Comp. Physiol. A* **196**(12), 927–938 (2010)
5. Rind, F.C., Bramwell, D.I.: Neural network based on the input organization of an identified neuron signaling impending collision. *J. Neurophysiol.* **75**(3), 967–985 (1996)
6. Blanchard, M., Rind, F.C., Verschure, P.F.M.J.: Collision avoidance using a model of the locust LGMD neuron. *Robot. Auton. Syst.* **30**(1), 17–37 (2000)
7. Yue, S., Rind, F.C.: Collision detection in complex dynamic scenes using an LGMD-based visual neural network with feature enhancement. *IEEE Trans. Neural Netw.* **17**(3), 705–716 (2006)
8. Stafford, R., Santer, R.D., Rind, F.C.: A bio-inspired visual collision detection mechanism for cars: combining insect inspired neurons to create a robust system. *BioSystems* **87**, 164–171 (2007)
9. Meng, H., Yue, S., Hunter, A., Appiah, K., Hobden, M., Priestley, N., Hobden, P., Pettit, C.: A modified neural network model for the lobula giant movement detector with additional depth

- movement feature. In: Proceedings of International Joint Conference on Neural Networks, pp. 14–19. Atlanta, Georgia (2009)
10. Guest, B.B., Gray, J.R.: Responses of a looming-sensitive neuron to compound and paired object approaches. *J. Neurophysiol* **95**(3), 1428–1441 (2006)
 11. Gabbiani, F., Mo, C., Laurent, G.: Invariance of angular threshold computation in a wide-field looming-sensitive neuron. *J. Neurosci.* **21**(1), 314–329 (2001)
 12. Gabbiani, F., Krapp, H.G., Koch, C., Laurent, G.: Multiplicative computation in a visual neuron sensitive to looming. *Nature* **420**, 320–324 (2002)
 13. Badia, S.B.i, Bernardet, U, Verschure, P.F.M.J.: Non-linear neuronal responses as an emergent property of afferent networks: a case study of the locust lobula giant movement detector. *PLOS Comput. Biol.* **6**(3), e1000701 (2010)

Chapter 15

Vision Based Motion Estimation of Obstacles in Dynamic Unstructured Environments

Andrei Vatavu and Sergiu Nedevschi

Abstract Modeling static and dynamic traffic participants is an important requirement for driving assistance. Reliable speed estimation of obstacles is an essential goal especially when the surrounding environment is crowded and unstructured. In this chapter we propose a solution for real-time motion estimation of obstacles by using the pairwise alignment of object delimiters. Instead of involving the whole 3D point cloud, more compact polygonal models are extracted from a classified digital elevation map and are used as input data for the alignment process.

Keywords Motion estimation · Polygonal map · Object contour · Iterative closest point · Driving assistance · Stereo-vision · Object delimiters

15.1 Introduction

In the context of Advanced Driver Assistance Systems, modeling static and dynamic entities of the environment is a key problem. The detection of moving traffic participants is an essential intermediate step for higher level driving technology tasks such as collision warning and avoidance, path planning or parking assistance. The problem of dynamic environment representation becomes even more difficult when the surrounding world is unstructured and heterogeneous, including the cases of crowded urban centers, traffic intersections or off-road scenarios. The representation component may be influenced by several factors: noisy measurements, occlusions, wrong data association or unpredictable nature of the traffic participants. In such

A. Vatavu (✉) · S. Nedevschi
Computer Science Department, Technical University of Cluj-Napoca,
26-28 G. Baritiu Street, Cluj-Napoca, Romania
e-mail: Andrei.Vatavu@cs.utcluj.ro

S. Nedevschi
e-mail: Sergiu.Nedevschi@cs.utcluj.ro

complex environments, a driver assistance system should be able to detect other moving traffic entities in real-time and at a high accuracy.

Usually, the classic approaches of dynamic obstacles detection and tracking consist in extracting a set of features from the scene and estimating the motion from their displacement. Current solutions can directly use 3D points [1], or they can track high level attributes such as 2D boxes or 3D cuboids [2], stixels [3], free-form polygonal models [4], object contours [5, 6] etc.

The dynamic obstacle modeling solutions can be classified by the nature of used sensors. The most common used sensors are vision based [2], laser [7, 8], sonar [9] or radar. The motion estimation techniques are also distinguished by the level at which the dynamic features detection is applied. Some of the existing methods rely on computing motion before generating a model [10, 11], while other methods are based on extracting some attributes and subsequently estimating their dynamic parameters [2, 4, 5].

Many of dynamic object detection solutions use intermediate representations as primary information. A common practice is mapping 3D information into occupancy grids [10], digital elevation maps [12] or octrees [9].

The data association and identifying correct correspondences steps play an important role in estimating the motion of the traffic entities. One of the widely used methods for model fitting in the presence of many data outliers is the RANSAC algorithm [13]. However, its accuracy depends directly on the number of used samples. This may lead to a high computational cost.

Direct matching solutions such as Iterative Closest Point (ICP) [14] algorithm are most common for vehicle localization and mapping [4]. In [15] the convergence performance for several ICP variants is compared. An optimized ICP method that uses a constant time variant for finding the correspondences is presented. In [4] a moving objects map is segmented by assuming that dynamic parts do not fulfill the constraints of the SLAM. However, the most of scan matching methods do not take into consideration the ego-motion parameters. The data association of objects in subsequent scans is hard to be achieved when the traffic participants or the ego vehicle moves at high speeds or when the measurement uncertainties are not taken into account.

We propose a solution of representing the dynamic environment in real-time by using the pairwise alignment of free-form delimiters and considering the advantages provided by a stereovision system, by inheriting the object information from the intermediate representation. Instead of registering the whole 3D point cloud, our method is based on extracting the most visible object cells from the ego car and using them as input data for the alignment process. We propose an extension of the classical ICP algorithm by applying a set of improvement heuristics:

- The data association is one of the problems of the classical scan matching techniques. It's hard to estimate the correspondent models from previous scans only based on the proximity criterion. In our case we introduce a pre-processing step. First, we find the correspondence pairs between the model set (contour extracted in previous frame) and the measurement set (current frame results) by finding

similarities between object blobs and passing this information at the contour level. Then, a list of associated contour candidates is generated and is used as the input for the next steps of the alignment;

- For the registration process we use free-form polygonal models that minimize the erroneous results caused by occlusions, or by stereo reconstruction errors. The main idea is that we are taking into account only the most visible points from the ego-vehicle by performing a radial scanning of the environment [16];
- The previously extracted speeds are used as the initial guess for the ICP algorithm;
- In order to filter the alignment outliers, a rejection metric that includes stereo uncertainties is proposed;

Our method is based on information provided by a Digital Elevation-Map, but can be easily adapted for other types of intermediate representations.

The remaining of the chapter is structured as follows: Sect. 15.2 introduces the architecture of the proposed dynamic environment representation. Section 15.3 presents the pre-processing module with a group of necessary tasks for extracting object dynamic properties. In Sect. 15.4, the main steps of the motion estimation component are detailed. The last two sections show the experimental results and conclusion about this contribution.

15.2 System Architecture

The dynamic environment representation method has been developed and adapted for crowded environments such as urban city traffic scenes. In this chapter we extend our previous Dense Stereo-Based Object Recognition System (DESBOR) [17]. The system architecture (see Fig. 15.1) could be divided in four main blocks: data acquisition and 3D reconstruction, intermediate representation, pre-processing, and motion estimation.

Data Acquisition and 3D Reconstruction is the first level of the processing flow. At this stage the images are acquired from the two cameras, then the 3D reconstruction is performed using a specialized TYZX [18] board. The resulted point cloud is used as the input information for computing the Digital Elevation Map.

Intermediate Representation: The raw dense stereo information is mapped into a Digital Elevation Map (see Fig. 15.2). The resulted intermediate representation contains three types of cells: road, traffic isle and object. The cells are labeled based on their height information. More details about the Elevation Map are presented in [19].

Pre-processing: The pre-processing level groups a set of basic tasks that are performed prior the ICP algorithm. At this phase, the object contours are extracted by radial scanning of the Elevation Map. For the delimiters extraction we use the Border Scanner algorithm previously developed by us [16]. We apply the ego-motion compensation for the Elevation Map and contours that are extracted in previous frame, assuming that we know the odometry information. The ego-vehicle motion is

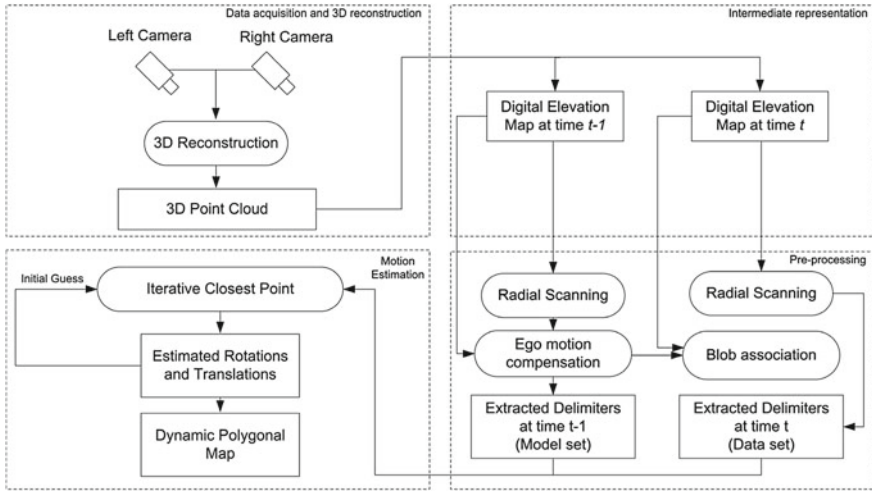
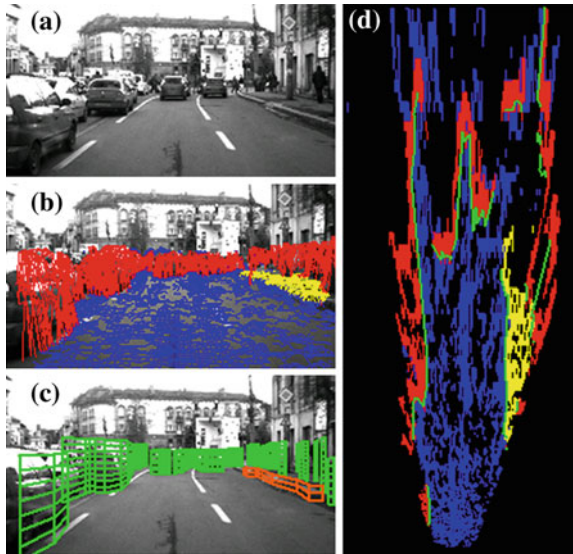


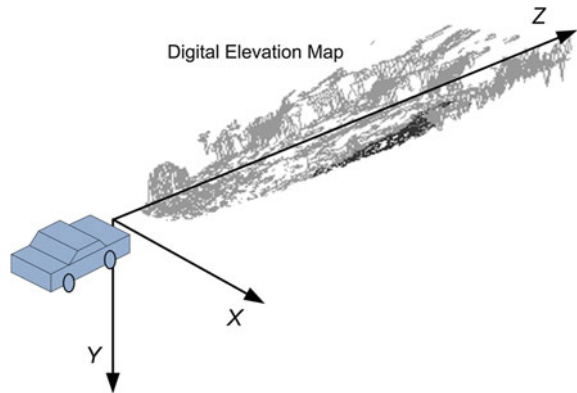
Fig. 15.1 System architecture

Fig. 15.2 a An urban traffic scene. b The elevation map projected on the left camera image. c A compact representation of the environment. d The top view of the elevation map. The elevation map cells are classified (blue—road, yellow—traffic isle, red—obstacles)



compensated in order to separate its speed from the independent motion of the objects in the traffic scene. Another pre-processing task is to associate the polygonal models. The data association is achieved by using the maximum overlapping score of the Elevation Map blobs. Considering that each polygonal model inherits the blob type, it also inherits the blob association information.

Motion Estimation: As the result of the pre-processing level, a list of candidates is provided for the ICP module. Each candidate represents a pair of associated contours

Fig. 15.3 Coordinate system

in the previous stage. For each candidate, a rotation and a translation is estimated by the ICP algorithm. Then the computed motion information is associated to the static polygonal models. A dynamic polyline map is generated as the result. Each polyline element is characterized by a set of vertices describing the polygon, position, height, type (traffic isle, obstacle), orientation and magnitude.

In our case the two cameras are placed on a moving vehicle. We use a coordinate system where the z axis points toward the direction of the ego-vehicle, and the x axis is oriented to the right. The origin of the coordinate system is situated in front of the car (see Fig. 15.3).

15.3 Pre-Processing Level

The pre-processing stage consists in performing necessary tasks prior the motion estimation. First, extracting a sufficiently generic model is needed. The extracted model should allow us the creation of fast subsequent algorithms and as well it should minimize the representation errors caused by noisy 3D reconstruction or by occlusions.

A second task is to separate the ego-vehicle speed from the independent motion of the other objects in the traffic scene. This is achieved by compensating the ego motion.

And finally, elevation map blob is labeled and is used in data association. As the result a list of pairs of contours is extracted and is provided subsequently to the ICP step. Thus, unlike the other classical methods that involve aligning the whole local maps at once, and then segmenting the dynamic obstacles from the static ones, we first associate the obstacles at the blob level and then apply the ICP for each associated candidate.

15.3.1 Polyline-Based Environment Perception

For the polyline based object representation we use the Border Scanner algorithm described by us in [16]. The main idea is that we are taking into account only the most visible points from the ego car and extract object delimiters by radial scanning of the Elevation Map. Our method is similar to a Ray-Casting approach. The proposed method consists in determining the first occupied point intersected by a virtual ray which extends from the ego-car position. The scanning axis moves in the radial direction, having a fixed center at the ego-vehicle position (the coordinate system origins). At each step we try to find the nearest visible point from the Ego Car situated on the scanning axis. In this way, all subsequent cells P_i are accumulated into a Contour List C , by moving the scanning axis in the radial direction:

$$C = \{P_1, P_2, \dots, P_n\} \quad (15.1)$$

For each object O_i described by a contour C_i we apply a polygonal approximation of C_i by using a split-and-merge technique described in [20]. The extracted polygon is used to build a compact 3D model based on the polyline set of vertices as well as on the object height. A polyline based representation is described in Fig. 15.2d.

15.3.2 Ego-Motion Compensation

Before estimating the motion of the traffic entities, the movement of the ego vehicle must also be taken into consideration. In order to compensate the ego motion in the successive frames, for each given point $P_{t-1}(x_{t-1}, y_{t-1}, z_{t-1})$ in the previous frame, the corresponding coordinates $P_t(x_t, y_t, z_t)$ in the current frame are computed by applying the following transformation:

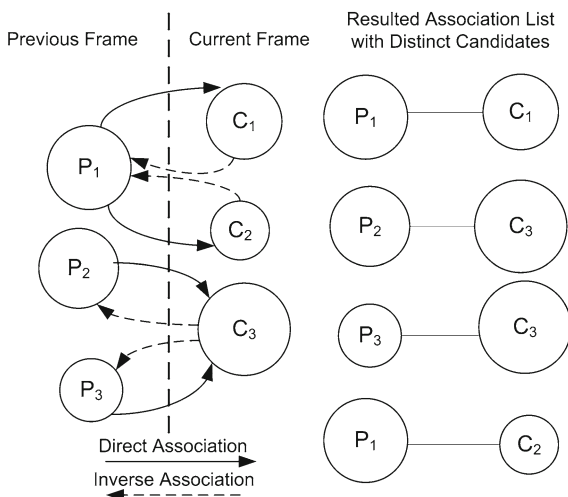
$$\begin{bmatrix} x_t \\ y_t \\ z_t \end{bmatrix} = R_y(\psi) \begin{bmatrix} x_{t-1} \\ y_{t-1} \\ z_{t-1} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ t_z \end{bmatrix} \quad (15.2)$$

where $R_y(\psi)$ is the rotation matrix around the Y axis with a given angle ψ , and t_z is the translation on the Z axis. The rotation and the translation parameters are provided by the ego-car odometry. It is considered that the translations on the X and Y axis are zero.

15.3.3 Data Association

This stage consists in finding the corresponding contours that identify a single object in consecutive frames. As each extracted contour describes an Elevation Map blob, finding the associated contour pairs is reduced to find a similarity between the object blobs.

Fig. 15.4 The association between two set of blobs in the consecutive frames and the resulting set of associated pairs



For each object P_i from the previous frame and for each object C_j from the current frame we calculate an overlapping score A_{ij} . The results are stored into a score matrix $A = \{A_{ij}\}$. Candidates with the highest score are taken into account in determining the associations between the two set of objects P and C .

However the association problem may lead only to partial results in the cases when larger objects from the previous frame are split into smaller blobs in the current frame and vice versa. In order to find all possible pairs of candidates we perform two types of associations: a direct association (forward association) finding best overlapping candidates in the current frame for all blobs in the previous frame, and a reverse association (backward association) that finds best overlapped objects in the previous frame for all objects from the current frame (see Fig. 15.4). The final list of candidates includes all distinct pairs associated in the two steps.

15.4 Motion Estimation

The object motion estimation module receives as input a list of associated contour pairs. For each distinct pair we compute correspondences between the two contours and estimate a rotation and a translation which minimize the alignment error. For the contour pairwise registration we use the Iterative Closest Point (ICP) method. The ICP algorithm was proposed by Besl and McKay [14] and represents a common solution especially for scan-matching techniques, but the idea could be adapted for any kind of models.

For each contour pair that identifies the same object in the consecutive frames we define two set of points: a model set $P = \{p_1, p_2, \dots, p_M\}$ that describes the object contour in the previous frame, and a data set $Q = \{q_1, q_2, \dots, q_K\}$ that describes

the object contour in the current frame. For each point q_j from Q the corresponding closest point p_i from P is found. We want to find an optimal rotation R and translation T that minimize the alignment error. The objective function is defined:

$$E(R, T) = \sum_{i=1}^N \|Rp_i + T - q_i\|^2 \quad (15.3)$$

where p_i and q_i are the corresponding point pairs of the two sets and N is the total number of correspondences.

The proposed alignment method is described by the following main steps:

1. **Matching**—for each point from data set, the closest point from the model set is found. A list of correspondent pairs is generated.
2. **Outliers Rejection**—Rejecting the outliers that could introduce a bias in the estimation of translation and rotation.
3. **Error Minimization**—estimating new transformation parameters R and T for the next iteration.
4. **Updating**—having the new R and T , a new target set is computed by applying the new transformation to the model set. A global transformation M_g is updated with the new R and T values.
5. **Testing the Convergence**—compute the average point-to-point distance between the measurement set and transformed model set. Then test if the algorithm has been converged to a desired result. If the error is greater than a given threshold, the process continues with a new iteration. The algorithm stops when the computed error is below the selected error threshold or when a maximum number of iterations have been achieved.

Next we will detail each of these steps.

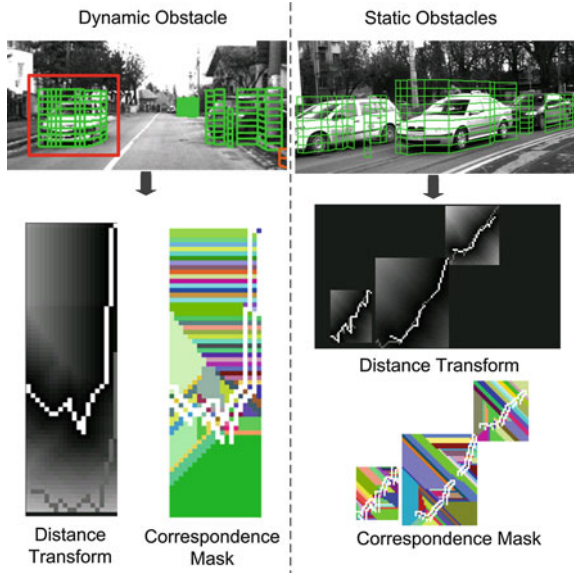
15.4.1 Matching

At this stage, for each point q_i from Q we want to find the closest point from the model set P :

$$d(q_i, P) = \min_{j \in \{1..N_p\}} d(q_i, p_j) \quad (15.4)$$

Usually this task is the most computationally extensive in the ICP algorithm. The classical brute force search approach has a complexity of $O(N_q \cdot N_p)$, with N_p being the number of points in P and N_q —the number of points in Q . In order to reduce the complexity to $O(N_q \cdot \log N_p)$ many solutions employ a KD-Tree [21] data structure. In our case, for finding closest points problem, we use a modified version of Chamfer based Distance Transform [22] (see Fig. 15.5).

Fig. 15.5 Distance transforms and corresponding masks are computed for dynamic obstacles (*left side*), and for static obstacles (*right side*). Data contours (*gray color*) and model contours (*white color*) are superimposed on the distance transform image. Each contour point in the correspondence mask is labeled with a unique color. The colors in the corresponding mask identify uniquely the closest contour point (having the same color)



A distance transform represents a map that has the property that each map cell has a value proportional to the nearest obstacle point. In our case, for each separate model contour we define a region of interest and compute the distance transform. The difference of our solution is that we use two maps: a distance map that stores the minimum distances to the closest points, and a correspondence map, storing the positions of the closest points (see Fig. 15.5). The correspondences from the model set are identified by superimposing the data contour on the two masks.

15.4.2 Outliers Rejection

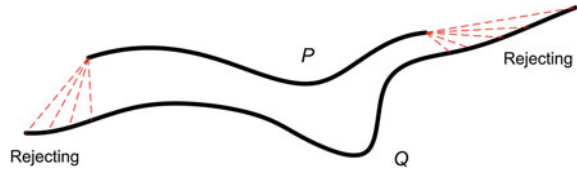
The purpose of this stage is to filter erroneous correspondences that could influence the alignment process. We use two types of rejection strategies: rejection of pairs whose point-to-point distance is greater than a given threshold, and eliminating the points where the overlap between the two contours is not complete.

15.4.2.1 Distance Based Rejection

The classical strategy consists in rejection of pairs whose point-to-point distance is larger than a given threshold D_t :

$$d(q_i, p_j) > D_t \tag{15.5}$$

Fig. 15.6 Rejecting the contour boundary



Because the stereo reconstruction error generally increases with the square of the z distance, the stereo-system uncertainties must be taken into account. As suggested by [10], if we assume that the stereo-vision system is rectified, then the z error is given by the following relation:

$$\sigma_z = \frac{z^2 \cdot \sigma_d}{b \cdot f} \tag{15.6}$$

where z is the depth distance, b is the stereo system baseline; f is the focal length and σ_d denotes the disparity error. Thus, for each corresponding pair of points (p_i, q_i) from the two sets, the rejection is made if:

$$d(q_i, p_j) > D_t + \sigma_z \tag{15.7}$$

This would mean that the rejecting threshold is increased at once with the z distance.

15.4.2.2 Boundary Based Rejection

The second type of rejecting is filtering the point correspondences caused by incomplete overlap between contours. Usually, these situations appear when one of the two contours is incompletely extracted due to occlusions, and may lead to incorrect alignments. A possible solution is to identify the subsets of points from Q that have the same correspondent point p_j in P , and keeping only the pair with the minimum distance (see Fig. 15.6).

15.4.3 Error Minimization

In this step we determine the optimal rotation R and translation T by minimizing the objective function defined by Eq. (15.3).

The rotation matrix around the Y axis is linearized, approximating $\cos \alpha$ by 1 and $\sin \alpha \approx \alpha$ by α :

$$R_y(\alpha) = \begin{bmatrix} \cos \alpha & 0 & \sin \alpha \\ 0 & 1 & 0 \\ -\sin \alpha & 0 & \cos \alpha \end{bmatrix} \approx \begin{bmatrix} 1 & 0 & \alpha \\ 0 & 1 & 0 \\ -\alpha & 0 & 1 \end{bmatrix} \quad (15.8)$$

The translation vector is defined as:

$$T = [t_x \ t_y \ t_z]^T \quad (15.9)$$

We can rewrite the Eq. (15.3) as:

$$E(R, T) = \sum_{i=1}^N \left\| \begin{bmatrix} 1 & 0 & \alpha \\ 0 & 1 & 0 \\ -\alpha & 0 & 1 \end{bmatrix} \begin{bmatrix} p_{i,x} \\ p_{i,y} \\ p_{i,z} \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} - \begin{bmatrix} q_{i,x} \\ q_{i,y} \\ q_{i,z} \end{bmatrix} \right\|^2 \quad (15.10)$$

The $E(R, T)$ is minimized with respect to α , t_x , t_y , and t_z by setting the partial derivatives to zero:

$$\begin{cases} \frac{\partial E(R, T)}{\partial \alpha} = 2 \sum_{i=1}^N \left[\alpha(p_{i,x}^2 + p_{i,z}^2) + t_{i,x} p_{i,z} - t_{i,z} p_{i,x} - q_{i,x} p_{i,z} + q_{i,z} p_{i,x} \right] = 0 \\ \frac{\partial E(R, T)}{\partial t_x} = 2 \sum_{i=1}^N (t_{i,x} + p_{i,x} + \alpha p_{i,z} - q_{i,x}) = 0 \\ \frac{\partial E(R, T)}{\partial t_y} = 2 \sum_{i=1}^N (t_{i,y} + p_{i,y} - q_{i,y}) = 0 \\ \frac{\partial E(R, T)}{\partial t_z} = 2 \sum_{i=1}^N (t_{i,z} + p_{i,z} - \alpha p_{i,x} - q_{i,z}) = 0 \end{cases} \quad (15.11)$$

Therefore we can obtain the unknown coefficients:

$$\begin{cases} \alpha = \frac{1}{N \sum_{i=1}^N (p_{i,x}^2 + p_{i,z}^2) - \left(\sum_{i=1}^N p_{i,x} \right)^2 - \left(\sum_{i=1}^N p_{i,z} \right)^2} \cdot \left(\sum_{i=1}^N p_{i,x} \sum_{i=1}^N q_{i,z} - \sum_{i=1}^N p_{i,z} \sum_{i=1}^N q_{i,x} + N \left[\sum_{i=1}^N (q_{i,x} p_{i,z}) - \sum_{i=1}^N (q_{i,z} p_{i,x}) \right] \right) \\ t_x = \frac{1}{N} \left(\sum_{i=1}^N q_{i,x} - \sum_{i=1}^N p_{i,x} - \alpha \sum_{i=1}^N p_{i,z} \right) \\ t_y = \frac{1}{N} \left(\sum_{i=1}^N q_{i,y} - \sum_{i=1}^N p_{i,y} \right) \\ t_z = \frac{1}{N} \left(\sum_{i=1}^N q_{i,z} - \sum_{i=1}^N p_{i,z} + \alpha \sum_{i=1}^N p_{i,x} \right) \end{cases} \quad (15.12)$$

15.4.4 Updating

Assuming that we have estimated new R and T parameters in the previous step, a new target set is computed by applying the new transformation to the model set.

Having the rigid body transformation matrix M :

$$M = \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & \alpha & t_x \\ 0 & 1 & 0 & t_y \\ -\alpha & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (15.13)$$

Each point p_i from the model set P is transformed according to the following relation:

$$\begin{bmatrix} p_{i,x}^{k+1} \\ p_{i,y}^{k+1} \\ p_{i,z}^{k+1} \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & \alpha & t_x \\ 0 & 1 & 0 & t_y \\ -\alpha & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_{i,x}^k \\ p_{i,y}^k \\ p_{i,z}^k \\ 1 \end{bmatrix} \quad (15.14)$$

Finally, a global transformation M_G is updated:

$$M_G^{k+1} = M_G^k \cdot M \quad (15.15)$$

15.4.5 Testing the Convergence

The error metric is estimated by computing the average Euclidean distance (AED) of every corresponding pair of data set Q and transformed model set.

$$Err = \frac{1}{N} \sum_{i=1}^N \|p_i - q_i\| \quad (15.16)$$

If the error is greater than a given threshold, the process continues with a new iteration. The algorithm stops when the computed error is below the selected error threshold or when a maximum number of iterations have been achieved.

15.5 Experimental Results

The proposed dynamic environment representation method has been tested in different traffic situations. For our experiment we used a 2.66GHz Intel Core 2 Duo Computer with 2GB of RAM. Figure 15.7 presents some qualitative results obtained in a dynamic urban traffic scenario.

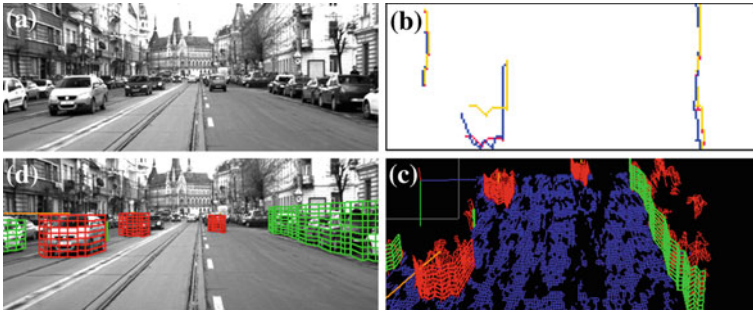


Fig. 15.7 **a** An urban traffic scenario. **b** The alignment result (*red color*) between the model delimiter extracted in previous frame (*yellow*) and the data contour, extracted in the current frame (*blue*). **c** The virtual view of the scene. The static obstacles are represented with *green* delimiters while the dynamic obstacles are colored with *red*. The speed vectors are associated to each dynamic entity. **d** The representation result, projected on the *left* camera image

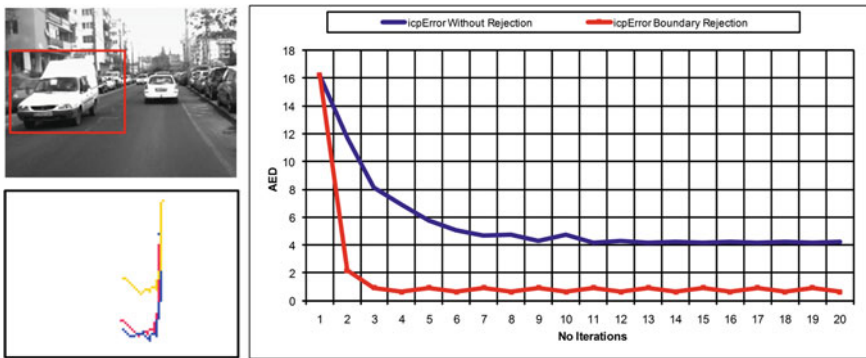


Fig. 15.8 The computed Error Metric in the case of ICP algorithm that does not use outlier rejection and ICP method that uses a boundary rejection

In Fig. 15.7b the model delimiter that was extracted in previous frame is colored with yellow, while the data contour (extracted in the current frame) is drawn with blue. The result of the alignment is illustrated with red color. It can be observed that in the case of the incoming vehicle, as well as for the lateral static vehicles, the aligned model is superimposed almost perfectly on the data set. In the Fig. 15.7c, the virtual view of the scene is shown. The static obstacles are represented with green delimiters while the dynamic obstacles are colored with red. The speed vectors are associated to the each dynamic entity (yellow color). The representation result is also projected on the left camera image (see Fig. 15.7d). We considered that the obstacles with a speed greater than 8 km/h are dynamic.

Figure 15.8 shows a comparative result between the ICP algorithm that includes all correspondence points (blue color) and the alignment method that uses the Contour Boundary Rejection strategy (red color). We used the Average Euclidean

Distance (AED) as the error metric. It can be observed that the ICP algorithm based on Boundary Rejection strategy converge more quickly than the ICP method without a filtering mechanism and proves to be more accurate having a lower alignment error. For our experiments we used a maximum number of 10 iterations. The average processing time was about 38 ms.

15.6 Conclusions

In this chapter we propose a method of real-time representation of the dynamic environment by using the pairwise alignment of free-form models. Instead of registering the whole 3D point cloud, the most visible obstacle points from the ego car are extracted and are subjected to the alignment process. We extend the classical ICP algorithm with a set of preprocessing tasks. First, we associate the delimiters at the blob level. Then, a list of associated candidates is passed to the alignment stage. For the registration process we use free-form polygonal models that minimize the erroneous results caused by occlusions, or by stereo reconstruction errors. As future work we propose to improve the stability of the environment perception by extending our system with a temporal filtering of the estimated speeds.

References

1. Franke, U., Rabe, C., Badino, H., Gehrig, S.: 6d-vision: Fusion of stereo and motion for robust environment perception. In: 27th Annual Meeting of the German Association for Pattern Recognition DAGM '05, pp. 216–223 (2005)
2. Danescu, R., Nedevschi, S., Meinecke, M.M., Graf, T.: Stereovision based vehicle tracking in urban traffic environments. In: IEEE Intelligent Transportation Systems Conference (ITSC 2007), Seattle, USA (2007)
3. Pfeiffer, D., Franke, U.: Efficient representation of traffic scenes by means of dynamic stixels. In: IEEE Intelligent Vehicles Symposium (IEEE-IV) 2010, pp. 217–224 (2010)
4. Wang, C.C., Thorpe, C., Hebert, M., Thrun, S., Durrant-Whyte, H.: Simultaneous localization, mapping and moving object tracking. *Int. J. Robot. Res.* **26**(6), 889–916 (2007)
5. Prakash, S., Thomas, S.: Contour tracking with condensation/stochastic search. In: Department of CSE, IIT Kanpur (2007)
6. Yokoyama, M., Poggio, T.: A contour-based moving object detection and tracking. In: 14th International Conference on Computer Communications and Networks (ICCCN '05), pp. 271–276, IEEE Computer Society Washington, DC, USA (2005)
7. Thomas de Candia: 3d tracking of dynamic objects with a laser and a camera. Technical report, Autonomous System Lab, ETH Zurich (2010)
8. Madhavan, R.: Terrain aided localization of autonomous vehicles. In: Symposium on Automation and Robotics in Construction, Gaithersburg (2002)
9. Fairfield, N., Kantor, G.A., Wettergreen, D.: Real-time SLAM with octree evidence grids for exploration in underwater tunnels. *J. Field Robot.* **24**, 3–21 (2007)
10. Danescu, R., Oniga, F., Nedevschi, S.: Modeling and tracking the driving environment with a particle based occupancy grid. *IEEE Trans. Intell. Transp. Syst.* **12**(4), 1331–1342 (2012)

11. Hess, J.M.: Characterizing dynamic objects in 3d laser range data. Master's thesis, Albert-Ludwigs-Universitt Freiburg (2008)
12. Danescu, R., Oniga, F., Nedeveschi, S., Meinecke, M.-M.: Tracking multiple objects using particle filters and digital elevation maps. In: IEEE Intelligent Vehicles Symposium (IEEE-IV 2009), pp. 88–93, Xi'an, China (2009)
13. Fischler, M., Bolles, R.: Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM* **24**(6), 381–395 (1981)
14. Besl P., McKay, N.: A method for registration of 3d shape. *Trans. Pattern Anal. Mach. Intell.* **12**(2), 239–256 (1992)
15. Rusinkiewicz, S., Levoy, M.: Efficient variants of the ICP algorithm. In: Third International Conference on 3D Digital Imaging and Modeling (2001)
16. Vatavu, A., Nedeveschi, S., Oniga, F.: Real time object delimiters extraction for environment representation in driving scenarios. In: ICINCO-RA 2009, pp. 86–93, Milano, Italy (2009)
17. Nedeveschi, S., Danescu, R., Marita, T., Oniga, F., Pocol, C., Sobol, S., Tomiuc, C., Vancea, C., Meinecke, M.M., Graf, T., To, T.B., Obojski, M.A.: A sensor for urban driving assistance systems based on dense stereovision. In: Intelligent Vehicles 2007, pp. 278–286, Istanbul (2007)
18. Woodill, J.I., Gordon, G., Buck, R.: Tyzx deepsea high speed stereo vision system. In: IEEE Computer Society Workshop on Real Time 3-D Sensors and Their Use, Conference on Computer Vision and, Pattern Recognition (2004)
19. Oniga, F., Nedeveschi, S.: Processing dense stereo data using elevation maps: road surface, traffic isle, and obstacle detection. *IEEE Trans. Veh. Technol.* **59**(3), 1172–1182 (2010)
20. Douglas, D., Peucker, T.: Algorithms for the reduction of the number of points required to represent a digitised line or its caricature. *Can. Cartographer* **10**, 112–122 (1973)
21. Bentley, J.L.: Multidimensional binary search trees used for associative searching. *Commun. ACM* **18**(9), 509–517 (1975)
22. Borgefors, G.: Distance transformations in arbitrary dimensions. *Comput. Vis. Graph. Image Proc.* **27**, 321–345 (1984)

Chapter 16

Real-Time Vision-Based Pedestrian Detection in a Truck's Blind Spot Zone Using a Warping Window Approach

Kristof Van Beeck, Toon Goedemé and Tinne Tuytelaars

Abstract In this chapter we present a vision-based pedestrian tracking system targeting a specific application: avoiding accidents in the blind spot zone of trucks. Existing blind spot safety systems do not offer a complete solution to this problem. Therefore we propose an active alarm system, which automatically detects vulnerable road users in blind spot camera images, and warns the truck driver about their presence. The demanding time constraint, the need for a high accuracy and the large distortion that a blind spot camera introduces makes this a challenging task. To achieve this we propose a warping window multi-pedestrian tracking algorithm. Our algorithm achieves real-time performance while maintaining high accuracy. To evaluate our algorithm we recorded several pedestrian datasets with a real blind spot camera mounted on a real truck, consisting of realistic simulated dangerous blind spot situations. Furthermore we recorded and performed preliminary experiments with datasets including bicyclists.

Keywords Computer vision · Pedestrian tracking · Real-time · Active safety systems

K. Van Beeck (✉) · T. Goedemé · T. Tuytelaars
EAVISE, KU Leuven - Campus De Nayer, J. De Nayerlaan 5, 2860 Sint-Katelijne-Waver, Belgium
e-mail: kristof.vanbeeck@esat.kuleuven.be

K. Van Beeck · T. Goedemé · T. Tuytelaars
ESAT/PSI - VISICS, IBBT, KU Leuven, Kasteelpark Arenberg 10, 3001 Leuven, Belgium
e-mail: toon.goedeme@esat.kuleuven.be

T. Tuytelaars
e-mail: tinne.tuytelaars@esat.kuleuven.be

16.1 Introduction

Research shows that in the European Union, each year an estimate of 1300 people die due to blind spot accidents [7]. This so-called blind spot zone, mainly situated to the right side of the truck, is defined as a zone in which the truck driver has no or limited view. Existing commercial systems appear unable to completely cope with the problem. Each type of system has its own specific disadvantages. Currently the most widely used solution is the blind spot mirror. Since the introduction of this mirror however, which is obliged by law in the EU since 2003, the number of casualties did not decrease [13]. This is mainly due to the fact that these mirrors are often deliberately adjusted incorrect to facilitate maneuvering. Another popular system is the blind spot camera, a wide-angle camera aimed at the blind spot zone (see Fig. 16.1), combined with a monitor in the cabin of the truck. The advantage of the latter system is that the camera is always adjusted correctly, since it is robustly mounted onto the truck's cabin. These two types of safety systems are called passive systems, since they depend on the attentiveness of the truck driver, whereas active safety systems automatically generate an alarm. An example of such an active system is found in ultrasonic sensors placed at the side of the truck. When using these kind of systems, the problem of scene interpretation arises. Since they cannot distinguish static objects (e.g. traffic signs or trees) and pedestrians, they tend to often generate unnecessary alarms. In practice the truck driver will find this annoying and turns the system off. To overcome these problems our final target is to develop an active blind spot camera system. This driver-independent system automatically detects vulnerable road users in the blind spot zone, and warns the truck driver about their presence. Such a system has a wide range of advantages as compared to the previous safety systems: it is always adjusted correctly, is independent of the interpretation of the truck driver and is easily implementable in existing passive blind spot camera systems. Building such a system is an extremely challenging task, since vulnerable road users are a very diverse class. They not only consist of pedestrians but also bicyclists, mopeds, wheelchair users and children are included. Besides the objects that need to be detected, the nature of this specific problem introduces another challenge: due to the position of the camera (which is aimed at the blind spot zone), we have a highly dynamical background. And since the camera is moving, standard computer vision techniques like adaptive background estimation or background subtraction, which can be computed very fast and would largely facilitate the detection task, are not an option. However, the biggest challenge is the hard real-time constraint of this application combined with the need for a high precision and recall rate.

In this chapter we present part of such a complete safety system: we developed a real-time robust multi-pedestrian detector/tracker for real blind spot camera images which maintains high accuracy. In the future we plan to extend our system to multi-class. As opposed to the classically used sliding window approach, our algorithm is based on a *warping window approach*. In previous work we performed initial blind spot pedestrian detection experiments using a standard camera, mounted on a standard car [15]. Here, we present our warping window approach to cope with the

Fig. 16.1 The blind spot zone of trucks often creates dangerous situations



Fig. 16.2 Example frame of our blind spot camera setup



specific viewing angle of a real blind spot camera mounted on a real truck, and the distortion that this camera introduces. An example frame of our blind spot camera setup is displayed in Fig. 16.2. One clearly sees that standard pedestrian detectors (discussed in the next section), even if they were fast enough, cannot be used on these images because they are developed for pedestrians that appear upright in the image. Using our framework we manage to robustly detect and track the pedestrians while maintaining excellent speed performance. This is briefly done as follows. Using our warping window method, we can warp the regions of interest in the image and use a standard pedestrian detector at only one specific scale, which is very fast. We then integrate this approach in a tracking-by-detection framework, and further speedup the algorithm using temporal information to reduce the search space. To meet the strict accuracy demands, we use a pedestrian detector [8] which has very good accuracy at the cost of high computation time when it is used as is. Using our framework this detector still achieves high accuracy but at real-time performance (on our dataset we achieve an average frame rate of 10 fps). Since to our knowledge no truck blind spot camera datasets are available in the literature, we recorded our own real-life datasets in which we simulated different dangerous blind spot scenarios using a real truck. These images are used to evaluate our algorithm regarding both to speed and accuracy. The outline of this chapter is as follows: Sect. 16.2 discusses related work on this topic. Section 16.3 describes our pedestrian tracking algorithm in detail. In Sect. 16.4 we describe the datasets that we recorded together with the result of our approach. We conclude in Sect. 16.5 with final remarks and future work.

16.2 Related Work

A vast amount of literature concerning pedestrian detection is available. In [1] the authors propose the use of Histograms of Oriented Gradients (HOG). This idea was further extended to a multi part-based model in [9]. Later these authors further optimized their detection algorithm, and introduced a cascaded version [8]. All of the mentioned detectors use a sliding window paradigm: across the entire image one tries to find pedestrians at all possible locations and scales. This approach does not achieve real-time performance at the moment. To overcome this problem methods have been proposed that use a detector cascade with a fast rejection of the false detections [16], whereas others methods use a branch and bound scheme [12]. To avoid the need to fully construct the scale-space pyramid Dollár et al. proposed a multiscale pedestrian detector (coined *The fastest pedestrian detection in the west* or FPDW) which uses feature responses computed at a specific scale to approximate features responses at scales nearby [2]. Several comparative works on pedestrian trackers exist in the literature. In [5] a comparison is given between the Dalal and Triggs model (HOG combined with a linear SVM classifier) with a wavelet-based AdaBoost cascade. Their work shows a clear advantage of the HOG-based approach at the cost of lower processing speeds. In [3] seven pedestrian detectors, all based on HOG or Haar features trained with a boosting method or SVMs are compared. They concluded that the HOG detectors perform best for unoccluded pedestrians over 80 pixels high. A multifeature combination (HOG combined with Haar features) outperforms HOG in more difficult situations at an evidently higher computational cost. More recently, in [4] the same authors present an exhaustive evaluation of sixteen state-of-the-art pedestrian detectors. Their evaluation shows that part-based pedestrian detectors still achieve the highest accuracy, while the FPDW is at least one order of magnitude faster with only minor loss of accuracy. These results were our motivation to use the part-based pedestrian detector [8] as a base detector in our framework. Regarding pedestrian tracking algorithms, most of them rely on a fixed camera, and use a form of background subtraction [14, 17]. As mentioned this cannot be used in our application, since we have to work with moving camera images. Due to the specific blind spot view, which is a backwards/sideways looking view, detecting and tracking pedestrians is not a trivial task. Existing pedestrian trackers on moving vehicles mostly use a forward-looking camera [6], thereby reducing the complexity of the scene. Often a stereo camera setup is used, and the disparity characteristics are exploited [10]. Since our goal is to develop a system which is easily integrated into existing blind spot camera systems we need to use a monocular approach. We differ from all of the trackers mentioned above: we aim to develop a monocular multi-pedestrian tracking system with field of view towards the blind spot zone of the vehicle at real-time performance, while maintaining high accuracy.

16.3 Pedestrian Tracking Algorithm

Our warping window algorithm is mainly based on the following observation. Looking at the blind spot camera example frame in Fig. 16.2 one clearly notices that, due to the specific position of the blind spot camera and the wide angle lens, pedestrians appear rotated and scaled. The crux of the matter is that the amount of rotation and scaling is only dependent on the position in the image. Thus, each pixel coordinate $\mathbf{x} = [x, y]$ represents a specific scale and pedestrian rotation. If at each pixel coordinate the corresponding rotation and scale is known, one can dramatically speedup pedestrian detection. Instead of a classic full scale-space search we can warp the region of interest, which is extracted based on the scale at that pixel coordinate, to upright pedestrians on one standard scale. This way we can use a standard pedestrian detector at only one scale, which is very fast. Besides our application, this approach can easily be generalized to other applications where such wide-angle distortion and/or non-standard camera viewpoints occurs (e.g. surveillance applications). To get the rotation and scale for each pixel coordinate a one-time calibration step is needed. To enable robust tracking we integrate this warping window approach into a tracking-by-detection framework. We use temporal information to predict the next pedestrian positions, eliminating the need for a full search over the entire image. The next subsections each describe part of the algorithm. First our warping window approach is described in detail. We then give a quantitative motivation for our pedestrian detector choice and the size of our standard scale in Sect. 16.3.2. The last subsection explains how we integrate our warping window approach into a robust tracking framework, and thus describes how our complete algorithm works.

16.3.1 Warping Window Approach

The warping window approach is visualized in Fig. 16.3. Given input images as in Fig. 16.2, the pedestrians appear rotated and scaled at different positions in the image. If we assume that we have a flat ground-plane, we know that the rotation and the scale of these pedestrians only depends on their position in the image. Thus if the scale s and rotation θ are known for each position in the image (visualized in the figure using the 2D *lookup functions* or LUF heat map plots), we can warp the pedestrian ROIs (I) into upright pedestrians at a standard scale (I_{warp}), using $I_{warp} = TI$, with transformation matrix T :

$$T = \begin{bmatrix} s \cos \theta & -s \sin \theta & t_x \\ s \sin \theta & s \cos \theta & t_y \\ 0 & 0 & 1 \end{bmatrix} \quad (16.1)$$

A one-scale detector is used to detect the pedestrians, and the output coordinates of the bounding boxes are retransformed into input image coordinates. These coordinates are then fed into our tracking framework, to determine the next pedes-

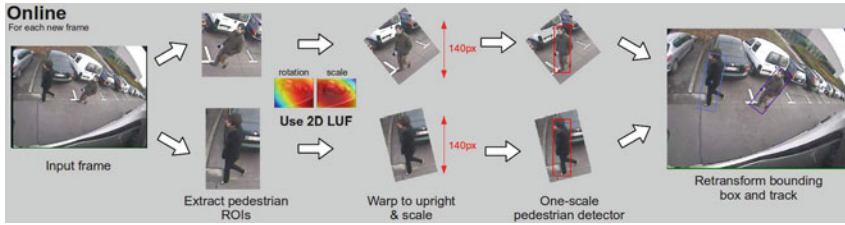


Fig. 16.3 Our warping window approach. If the scale and rotation are known, we can warp the ROIs and use a standard pedestrian detector at only one scale

trian ROIs. To determine the scale and rotation for each pixel coordinate, a one-time calibration step is needed. To achieve this, we manually labeled about 100 pedestrians in the calibration images homogeneously spread over the total image region. Each pedestrian yields scale and rotation data at that position. Next we fitted a two-dimensional second order polynomial function through the data points: $rotation = f_r(x, y)$ and $scale = f_s(x, y)$ where:

$$f_i(x, y) = A + Bx + Cy + Dx^2 + Exy + Fy^2 \quad (16.2)$$

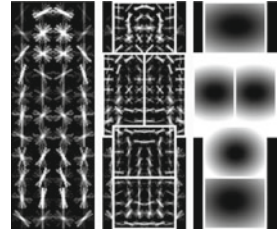
These two functions effectively represent a 2D lookup function, i.e. for each pixel coordinate they give the rotation and scale of that pixel position. If the camera position is adjusted, we need to perform a recalibration. However, due to the robust camera mounting on the truck this occurs only rarely.

Thus detecting pedestrians is composed of four steps: extract the pedestrian ROI, calculate the scale and rotation for that ROI, retransform to an upright pedestrian with a standard height of 140 pixels and use a pedestrian detector at only one scale. The choice for this number will be argued in the next subsection.

16.3.2 Pedestrian Detector

Since we only need to detect pedestrians at a standard scale (140 pixels), our approach allows the use of a detector with high accuracy which would otherwise be too computationally expensive. Given the extensive comparison results from [3–5] that we discussed in Sect. 16.2, two pedestrian detectors are applicable in our framework. Both the part-based detector [8, 9] and the FPDW [2] achieve high accuracy. The accuracy of the part-based models is slightly higher at the cost of a higher computation time due to scale-space pyramid construction. Since no scale-space pyramid needs to be constructed in our application, our choice evidently goes to the part-based detector. Let us now briefly discuss how this pedestrian detector works if used out-of-the-box. The object that has to be detected is described using a HOG model. The model consists of a root filter, representing the pedestrian appearance, and a

Fig. 16.4 The pedestrian HOG model. Root filter (*Left*), Part Filters (*Center*), Prior estimate of position of the part filters (*Right*)



number of smaller part filters, representing the head and limbs of the pedestrian (see Fig. 16.4). The position of each of the parts are latent variables, which are optimized during the detection. A first step is the construction of a scale-space pyramid from the original image. This is done by repeated smoothing and subsampling. For each entry of this pyramid, a feature map is computed, which is built using a variation of the HOG features presented by Dalal and Triggs [1]. For a specific scale one computes the response of the root filter and the feature map, combined with the response of the part filters and the feature map at twice the resolution at that scale. The transformed responses of the part filters are then combined with the response of the root filter to calculate a final score.

As a reference, if used out of the box on our images (640×480 resolution) this detector needs an average of 5.2s per frame (evaluated on a Intel Xeon Quad Core running at 3 GHz, all implementations are CPU-based only). If we reduce the number of scales to only contain those needed in our application, detection time decreases to about 850 ms. Later the authors presented their cascaded version [8]. There, using a weak hypothesis first, a fast rejection is possible while maintaining accuracy. Using this detector, again out of the box and only on the scales needed in our application, the detection time on our images equals 340 ms.

We altered both the default and the cascaded part-based pedestrian detector to a one-scale detector. In Fig. 16.5 (left) the average calculation times of the four different implementations, namely the part-based model with reduced scales (further referenced as *Felzenszwalb reduced scales*), our one-scale implementation of this detector (referenced as *Felzenszwalb one scale*), the cascaded version and our one-scale implementation of the cascaded version. Needless to say, the detection time strongly depends on the image resolution. To generate Fig. 16.5 (left), we used a high resolution pedestrian image and cropped the image to only contain the pedestrian. This image was then subsampled to the indicated resolutions. Calculation times are averaged over ten runs. Note that to obtain a fair comparison we deliberately did not cache any data. For example, the pedestrian model is completely reloaded into memory on each run. We can clearly see that decreasing the resolution drastically reduces the calculation time for both the standard Felzenszwalb and the cascaded detector. The calculation time of our one-scale implementations does decrease with resolution, but not nearly that fast. Since only one scale is looked at, a double gain in speed is realized. The scale-space pyramid does not need to be constructed, and features only need to be calculated and evaluated at one scale. In our warping window framework we use the cascaded one-scale detector.

Fig. 16.5 *Left* The calculation time for the different pedestrian detector implementations. *Right* The accuracy of our one-scale cascade detector implementation in function of the pedestrian resolution

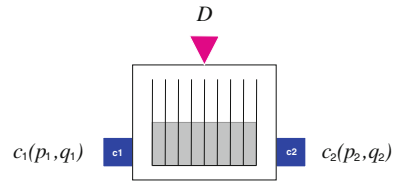


Fig. 16.6 Example of three initial search coordinates, together with the search regions that they define



Reducing the resolution implies that the accuracy significantly drops. Therefore we needed to determine the optimal trade-off point between the detection accuracy and the resolution to which we warp our pedestrian images. To determine that optimal resolution we extracted about 1000 pedestrians from our dataset, rescaled them to fixed resolutions and determined the accuracy of our one-scale cascaded detector for each resolution. These results are displayed in Fig. 16.5 (right). At higher pedestrian resolutions the accuracy remains almost constant at around 94 %. When decreasing the pedestrian resolution the accuracy starts to drop at approximately 135 pixels. Based on these observations we chose to rescale our pedestrians to a constant standard height of 140 pixels in our warping window approach. This results in an average calculation time of 45 ms when using the one-scale cascaded detector. If the model does not need to be reloaded on each run, calculation time further decreases to about 12 ms.

16.3.3 Tracking Framework

Our complete pedestrian tracking-by-detection algorithm works as follows. We integrate our warping window approach into a reliable tracking-by-detection framework. At positions where pedestrians are expected to enter the blind spot zone in the frame, standard search coordinates are defined, see Fig. 16.6. Our warping window approach is used to detect pedestrians at these search locations. If a pedestrian is detected, tracking starts. We use a linear Kalman filter [11] to estimate the next position of the pedestrian, based on a constant velocity model. Our experiments show that this assumption holds and suffices for a robust detection. We define the state vector x_k using the pixel position and velocity of the centre of mass of each



Fig. 16.7 Example output of our tracking algorithm

pedestrian: $x_k = [x \ y \ v_x \ v_y]^T$. The Kalman filter implements the following time update equation $\hat{x}_k^- = A\hat{x}_{k-1}$. Note that \hat{x}_k^- refers to the *a priori* state estimate at timestep k , while \hat{x}_k refers to the *a posteriori* state estimate at timestep k . The process matrix A equals:

$$A = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (16.3)$$

Using this motion model we predict the position of the pedestrian in the next frame. Around the estimated pedestrian centroid a circular region with a radius based on the scale at that coordinate, determined from the 2D scale LUF, is computed. In subsequent frames we use the estimated centroids and the standard search coordinates as inputs for our warping window approach. For each estimated centroid our warping window approach warps this ROI and seeks a new pedestrian detection. For every pedestrian that is being tracked, the algorithm evaluates if a centroid of such a new detection is found in its circular region. If a matching centroid is found, that Kalman filter is updated, and a new position is predicted. When multiple centroids are found, the nearest one is chosen. If for a tracked pedestrian no new detection is found, the Kalman filter is updated based on the estimated position. This enables tracking of partially occluded pedestrians or pedestrians where the HOG response is temporarily lower (e.g. because of background objects). When no new matching detection is found for multiple frames in a row (4 in our experiments), the tracker is discarded. If a detection is found with no previous tracked instance, tracking starts from there on. This approach eliminates the need for a full frame detection, thus limiting processing time. Figure 16.7 shows the output of our warping window tracking algorithm on one video sequence.

16.4 Experiments and Results

Due to the specific viewing angle of the blind spot camera no image datasets are available in the literature. Therefore we constructed such a dataset, consisting of several simulated dangerous blind spot scenarios. This was done using a real blind spot camera, mounted on a real truck. We used a commercial blind spot camera

Fig. 16.8 Our test truck (*Left*) with the mounted blind spot camera circled in red (*Right*)



(Orlaco CCC115°), which outputs 640×480 images at 15 frames per second. It has a wide-angle lens with a viewing angle of 115° . Figure 16.8 indicates the position of the blind spot camera on our truck. We recorded five different scenarios. At each scenario the truck driver makes a right turn, and the pedestrians react differently. For example, in some of the scenario's the truck driver takes a right turn while stopping to let the pedestrians cross the street, while in other scenario's the pedestrians stand still at the very last moment while the truck continues his turn. These simulations resulted in a dataset of about 11000 frames. Furthermore we recorded a second dataset which includes bicyclists, and consists of about 9000 frames. Our evaluation hardware consists of an Intel Xeon Quad Core, which runs at a clockspeed of 3 GHz. All implementations are CPU-based, we do not use GPU implementations. The algorithm is mainly implemented using Matlab, while part of the pedestrian detector is implemented in standard C-code. The image warping is implemented in OpenCV, using *mexopencv*. As mentioned in Sect. 16.3, as a reference, when used out of the box the Felzenszwalb pedestrian detector needs 5.2 s for a full scale-space detection over an entire frame. As our goal is to develop a real-time pedestrian tracker with high accuracy, we evaluated the algorithm with respect to both speed and accuracy. Our results are presented in the next subsections.

16.4.1 Speed Analysis

For each tracked pedestrian we need to do a new detection in the consequent frames. Thus if more pedestrians enter the frame, the total calculation time increases. Figure 16.9 (left) displays the detection time per tracked pedestrian in function of the rotation. We split up the total detection time in three separate steps: first the image is warped in an upright fixed scale pedestrian image. Then our pedestrian detector calculates the HOG features. The last step consists of the actual model evaluation, in which the image is given a score based on the HOG model. The total detection time increases if the rotation angle increases. Warping the window is computationally the least expensive operation. It only slightly depends on the rotational value, and maximally takes about 3 ms. The feature calculation and the model evaluation take almost an equal amount of time, and both increase with increasing rotation. This is due to the fact that the total image area increases with increasing rotation. If no rotation is needed, both feature calculation and model evaluation time take about 5 ms, resulting

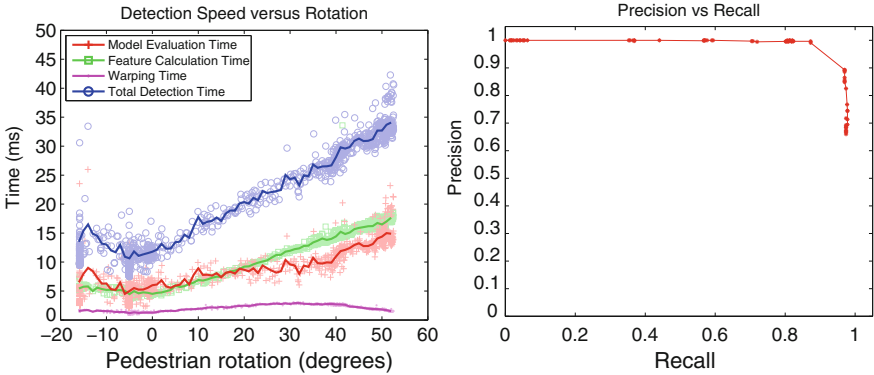


Fig. 16.9 *Left* Speed analysis of our warping window approach. The blue line indicates the total calculation time per pedestrian, in function of the rotation. *Right* A precision-recall graph of our algorithm as evaluated over our dataset

Table 16.1 Speed results as measured over our dataset

	Best-case	Average	Worst-case
FPS	50.8	10.1	7.8
# pedestrians	0	3.1	5

in a total detection time of 12 ms. In the worst-case scenario, occurring at a rotation of 52° (the maximum rotation in our application), the detection time increases to 35 ms. Thus if e.g. two pedestrians are tracked, of which one at low rotation and one at high rotation, detection time for these pedestrians requires about 45 ms. If two standard search regions are included at e.g. 15 ms each the total frame detection time equals 72 ms. In that case the algorithm achieves a frame rate of 14 frames per second. If multiple pedestrians are detected, detection speed decreases. Large groups of pedestrians are however easily noticed by the truck driver and therefore do not pose a real risk for accidents. Most blind spot accidents occur when only a few (mostly only one) pedestrian are in the blind spot zone. If only one pedestrian is tracked our algorithm achieves a frame rate of more than 20 frames per second. Table 16.1 shows the average, best-case and worst-case frame rate as evaluated over our dataset, and gives the number of pedestrians that were tracked while achieving these frame rates. Since in our dataset on average more than 3 pedestrians were visible per frame, the average calculation time given here is in fact an overestimation of the calculation time for a real scenario.

16.4.2 Accuracy Analysis

The accuracy of our detector is displayed in the precision-recall graph in Fig. 16.9 (right). They are determined as: $precision = \frac{TP}{TP+FP}$ and $recall = \frac{TP}{TP+FN}$. For each



Fig. 16.10 Example output of our tracking algorithm on a bicyclist dataset

pedestrian that our algorithm detects, we look for the centroid of a labeled pedestrian in the circular region of the detection. If this is the case, the detection is counted as a *true positive*. If no labeled pedestrian is found, the detection is indicated as being a *false positive*. If a labeled pedestrian is not detected, this is indicated as being a *false negative*. Our testset consists of about 1000 pedestrians in very diverse poses and movements. As can be seen in the figure our algorithm achieves both high precision and recall rates. At a recall rate of 94%, we still achieve a precision rate of 90%. This is due to the fact that using our warping window approach, the specific scale at each position is known. Therefore false positives are minimized, while the pedestrian detection threshold can be set very sensitive. This way difficult to detect pedestrians can still be tracked. While very good, the accuracy is not perfect yet. Our warping window approach sometimes fails to track pedestrians due to low responses of the HOG filter, induced because only a subtle intensity difference between the pedestrian and the background occasionally occurs. A possible solution for this is the inclusion of other features, e.g. motion information.

16.4.3 Preliminary Bicycle Experiments

An evident extension of our algorithm is the inclusion of other vulnerable road users. Here, as a first step towards a complete safety system, we present preliminary qualitative experimental results that we obtained with our algorithm on a bicyclist dataset. To conduct these experiments no algorithmic changes were performed. Our motivation for conducting these experiments is based on the fact that pedestrians and bicyclists share similar appearance features (e.g. the upperbody). Figure 16.10 shows the qualitative experimental results. As one can see in the first frames (frame 255 and 263), the appearance of the bicyclist is similar to that of a pedestrian. Therefore our algorithm performs well in these situations where the bicyclist is relatively far away. However, in consecutive frames the similarity decreases and the detection is lost. A merely direct application of the pedestrian detector is not feasible anymore. In the future, we plan to solve this problem by integrating other information cues, e.g. color histograms, to enhance the robustness of the detector. Another approach could be the retraining of the object detector to such specific appearance models.

16.5 Conclusions and Future Work

We presented a multi pedestrian tracking framework for a moving camera based on a warping window approach. We introduced this warping window approach to cope with the specific wide-angle distortion induced by the blind spot camera. Moreover, this methodology is easily applicable to other object detection applications in situations where such distortion occurs, e.g. caused by non-standard camera viewpoints or specific lenses. To evaluate our algorithms we recorded a representative real blind spot dataset. Experiments were performed evaluating both the speed and accuracy of our approach. Our algorithm achieves real-time performance while maintaining both high precision and recall. Furthermore we performed initial qualitative experiments on a bicyclist dataset. In the future we plan to extend our algorithm to refine the accuracy on other vulnerable road users classes such as bicyclists, mopeds, children and wheelchair users. We also plan to investigate if the inclusion of other information cues, e.g. motion features extracted from optical flow information, further increases the robustness of our detector.

References

1. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: International Conference on Computer Vision and Pattern Recognition, vol. 2, pp. 886–893 (2005)
2. Dollár, P., Belongie, S., Perona, P.: The fastest pedestrian detector in the west. In: Proceedings of the British Machine Vision Conference, pp. 68.1–68.11 (2010)
3. Dollár, P., Wojek, C., Schiele, B., Perona, P.: A benchmark. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Pedestrian detection (2009)
4. Dollár, P., Wojek, C., Schiele, B., Perona, P.: Pedestrian detection: An evaluation of the state of the art. *IEEE Trans. Pattern Anal. Mach. Intell.* **99** (2011)
5. Enzweiler, M., Gavrila, D.M.: Monocular pedestrian detection: survey and experiments. *IEEE Trans. Pattern Anal. Mach. Intell.* **31**(12), 2179–2195 (2009)
6. Ess, A., Leibe, B., Schindler, K., Gool, L.V.: A mobile vision system for robust multi-person tracking. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2008)
7. EU: Commission of the european communities, european road safety action programme: mid-term review (22 february 2006)
8. Felzenszwalb, P., Girschick, R., McAllester, D.: Cascade object detection with deformable part models. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2010)
9. Felzenszwalb, P., McAllester, D., Ramanan, D.: A discriminatively trained, multiscale, deformable part model. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2008)
10. Gavrila, D., Munder, S.: Multi-cue pedestrian detection and tracking from a moving vehicle. *Int. J. Comput. Vision* **73**(1), 41–59 (2007)
11. Kalman, R.: A new approach to linear filtering and prediction problems. *Trans. ASME J. Basic Eng.* **82**, 35–45 (1960)
12. Lampert, C., Blaschko, M., Hoffmann, T.: Efficient subwindow search: A branch and bound framework for object localization. *IEEE Trans. Pattern Anal. Mach. Intell.* **31**, 2129–2142 (2009)

13. Martensen, H.: Themarapport vrachtwagenongevallen 2000–2007 (BIVV) (2009)
14. Seitner, F., Hanbury, A.: Fast pedestrian tracking based on spatial features and colour. In: Proceedings of the 11th Computer Vision Winter Workshop (2006)
15. Van Beeck, K., Goedemé, T., Tuytelaars, T.: Towards an automatic blind spot camera: robust real-time pedestrian tracking from a moving camera. In: Proceedings of the twelfth IAPR Conference on Machine Vision Applications, pp. 528–531 (2011)
16. Viola, P., Jones, M.: Rapid object detection using a boosted cascade of simple features. In Proceedings of the IEEE Conference on Computer Vision and, Pattern Recognition, pp. 511–518 (2001)
17. Viola, P., Jones, M., Snow, D.: Detecting pedestrians using patterns of motion and appearance. *Int. J. Comput. Vision* **63**, 153–161 (2005)

Chapter 17

A Proposal of Risk Indexes at Signalised Intersections for ADAS Aimed to Road Safety

Bruno Dalla Chiara, Francesco Paolo Deflorio and Serena Cuzzola

Abstract Statistical data show that road intersections are one of the critical areas for accidents. The analyses reported are aimed at estimating risk indexes, which might be provided on-board, when vehicles approach a road intersection regulated by traffic lights, by an ADAS based on the use of the infrastructure-to-vehicle (I2V) or vehicle-to-infrastructure (V2I) communication systems. Two possible uses of the risk indexes can be identified: if data can be detected in real time, the driver could be informed on-board of a potentially hazardous situation, using algorithms to predict the dynamics of the vehicle on the basis of the data detected from the monitoring; the other use would be detecting—in case the vehicle were already within the dilemma zone—the lowest risk manoeuvre and providing the driver with a message on board. The chapter also reports the effects which might be generated by this ADAS application.

Keywords ADAS · Safety index · Intersection safety · Driver reaction time

17.1 Introduction

Quantifying the road safety risk and the effects that the Advanced Driver Assistance Systems (ADAS) can generate on it—i.e. the combined value of reducing the likelihood of an accident and its severity—is a very sensitive theme, which can today resort to the infrastructure-to-vehicle and/or vehicle-to-vehicle communication systems. Together with the interest towards the quality and energy efficiency of transport, safety is contributing to the fast spreading of Intelligent Transport Systems (ITS),

B. Dalla Chiara (✉) · F. P. Deflorio · S. Cuzzola
Department DIATI—Transport Engineering, Politecnico di Torino,
C.so Duca degli Abruzzi, 24-10129 Torino, Italy
e-mail: bruno.dallachiara@polito.it

F. P. Deflorio
e-mail: francesco.deflorio@polito.it

which include—amongst the different technologies—the ADAS. Such perspective is part of the migration—which has been in progress for years—from the mere passive safety to the study of systems, tools and applications which can ensure active, preventive and post-crash safety.

Within this context, attention is progressively concentrating on the cooperative systems, which can interact to one another, thus setting up communication between the different vehicles (Vehicle-to-Vehicle, V2V) or between the vehicles and the infrastructures (Vehicle-to-Infrastructure, V2I, or Infrastructure-to-Vehicle, I2V) to create ad hoc communication networks. This chapter focuses on the effects of integrating such communication systems with the ADAS with the aim of improving road safety; special attention is paid to the safety of the road users, in order to reduce both the number and the severity of the road accidents. The communication technologies between vehicles and between infrastructure and vehicles are suitable to intervene at the pre-crash stage, i.e. in emergency-assistance, where the action of the driver could still prevent the accident or reduce its risk.

It worth reminding a basic definition of crash, slightly reviewing the one which was proposed in [6]: the crash phase of an accident occurs when the perception-reaction time of a driver plus the time necessary to actuate the procedure (e.g., braking or steering) of the vehicle he/she is driving is greater than or equal to the time involved by the exogenous variation that occurs outside the vehicle; such a perception-reaction time of the driver plus that of the vehicle is therefore the maximum time available for the driver to respond to an emergency condition on the road and prevent an accident.

The road intersections are often considered as critical areas for the occurrence of crashes, because they increase the likelihood of the confluence of traffic streams from and to different directions.

17.2 State of the Art

During the last years, car manufacturers and researchers experimented many ADAS [17, 18]. These systems are in-vehicle technologies that provide support to various aspects of the driving task and they are supposed to improve traffic safety and traffic efficiency. In this field, the most famous and deployed ADAS systems are the adaptive cruise control (ACC) and the intelligent speed adaptation (ISA), collision avoidance systems, adaptive light control, lane departure warning, driver vigilance monitoring, pre-crash vehicle preparation and parking aid [19, 23].

In order to estimate the future impact of the ADAS development process from its very early stages, some studies were based on the use of microscopic traffic simulation. Torday et al. [24] proposed to integrate the output of this tool with a safety indicator, evaluated during the micro simulation process. The microscopic level of traffic description grants the opportunity of knowing the relative position of the vehicles, their speed and deceleration. All of these parameters thus enable the computation of a safety indicator useful to compare scenarios where ADAS are activated for vehicles. Other authors [20] provide an overview of micro-simulation

modelling for road safety impact assessment of ADAS. Recent literature and expert opinions identify driver behaviour sub-models and road safety indicators as key components. In Benz et al. [2], several existing models—on both the micro and macro scales—would be adapted and used to assess safety related effects of ITS measures. Examples of such measures include but are not limited to ADAS and IVIS (In-vehicle information systems). While the micro-models would determine the individual vehicles' safety related behaviour, the macro-models would investigate the network-wide aspects.

In order to enhance the performance of micro simulator for safety analysis, a Surrogate Safety Assessment Model (SSAM) has been developed [25]. This technique combines micro simulation and automated conflict analysis, which analyses the frequency and type of narrowly averted vehicle-to-vehicle collisions in traffic, to assess the safety of traffic facilities, without waiting for a statistically valid number of crashes and injuries to actually occur. Applications of this method to road intersection scenarios are reported in [9, 13, 15]. An assessment of the driver behaviour at the dilemma zone [16] and of the effectiveness of safety indicators based on the traffic conflict technique at intersection is reported in [1, 10].

Recent international research projects have been investigating both vehicle-based and road-based monitoring. The European projects SAFESPOT, COOPERS, CVIS and COVEL aimed at improving road safety by using intelligent vehicles interconnected to each other through a vehicular ad-hoc network (VANET).

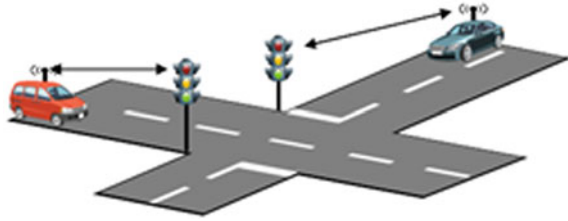
As regards the V2V and V2I communication systems and their relationships with safety and ADAS, they are a primary means for supplying information to drivers. In recent years, V2V and V2I communication systems have been submitted to intensive studies, also applied to safety at intersections (INTERSAFE-2).

In this field, FOTSIS was a large-scale field testing of the road infrastructure management systems needed for the operation of seven close-to-market cooperative I2V, V2I and I2I technologies (the FOTSis Services), which allowed assessing in detail both their effectiveness and their potential for a full-scale deployment in European roads.

We need to recall that the response time of a driver can be split into a mental processing and reaction time and a muscular time. The former includes the time from the perception of the external stimulus to the brain's message to the foot to brake. This implies the awareness of the hazard, the emotive response and the reaction itself. The muscular reaction time is needed for the right foot to move onto the brake pedal. The driver's reaction time is influenced by quick or slow reflexes, by his/her experience as well as by the complexity of the dangerous scenario that has to be faced. On the basis of tests and literature, the median perception-reaction time of a driver results to be 0.66 s, measured under normal highway driving conditions, with some degree of braking expectation, since the drivers were expecting the event to happen. From the moment the driver puts his/her foot onto the pedal, almost 0.1 s (inertia of the system) before the brake starts operating; this value may increase to 0.4 s in the case of slow and older braking equipment.

The diagram reported in [12], as well as on ISO technical standards and in [6], in revised editions, represents the distribution of a driver's brake perception-reaction

Fig. 17.1 Evocative image of I2V and V2I at a generic intersection regulated by traffic lights (INTERSAFE-2)



time between 0.2 and 2.1 s. The 95th percentile of perception-brake response times for these same conditions was 2.0". The findings from this study are consistent with the relevant literature: most drivers are capable of responding to an unexpected incident in 2.0" or less. Thus, the perception-reaction time of 2.5", adopted by the American Association of State Highway and Transportation Officials for design reasons, encompasses most of the driving population.

A driver who might need 0.3" of perception-reaction time under alerted conditions might need 1.5" under normal conditions; such response time may decrease by approximately 1" or more in an expected situation: IVC warning systems allow one to pass from an unexpected to an anticipated situation, and thus influence the perception-reaction time.

17.3 Safety Analysis at Intersections

This paragraph shows the results of the analyses developed on the effects and benefits which would be potentially generated by the forthcoming use of the infrastructure-vehicle (I2V) or vehicle—infrastructure (V2I) communication systems at the road intersections regulated by traffic lights (Fig. 17.1): a theme that—as it has been highlighted—is extensively being dealt within the literature.

The effects of the use of the I2V systems are assessed through the proposal of indicators on the likelihood and/or severity of the risk, which can timely and preventively indicate potentially critical conditions and send more or less intensive alarm messages—depending upon the criticality—on board the vehicles which are potentially involved by means of the I2V communications.

For the sake of completeness of the analysis, some proposals for the combined use of sensors to monitor the vehicles which approach the intersections [26] have been developed in our research.

Our analyses assume that the use of I2V systems would match the increased level of attention of the driver and—consequently—the dampening of the perception-reaction-actuation time ($t_{p,r,a}$) of the driver, with the subsequent increased safety margin.

The processing concerns the study of the driver's behaviour when the yellow light is triggered. All such processes associate the use of the I2V technologies to the

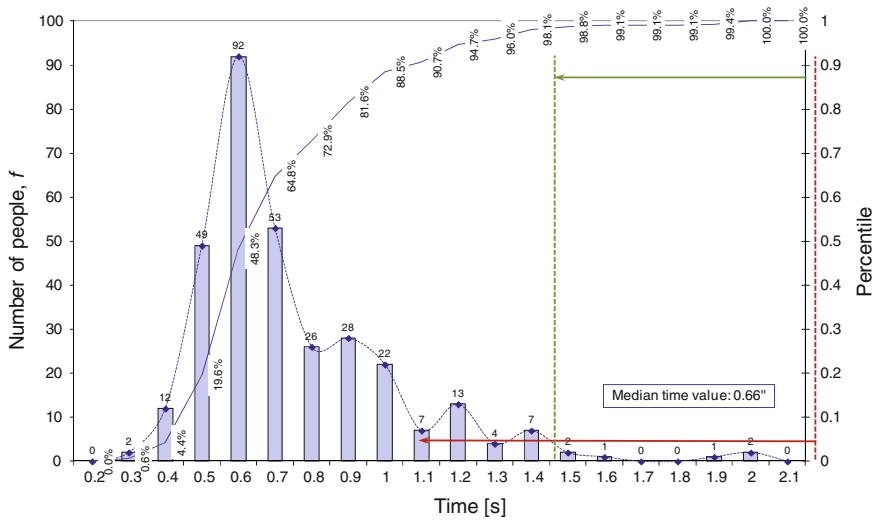


Fig. 17.2 Johansson and Rumar’s distribution of the $t_{p,r}$ highlighting the two values of $t_{p,r}$ which give $t_{p,r,a} = 1.46$ s (green arrow) and 1.1 s (red arrow)

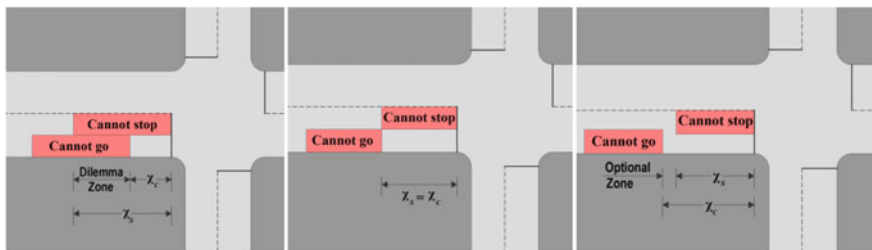


Fig. 17.3 Graphic representation of the zone where the vehicle cannot stop in safety conditions (*Cannot Stop*) or cannot clear the intersection in full safety conditions (*Cannot Go*)

maximum perception—reaction—actuation time of the driver ($t_{p,r,a}$) which—in case of low levels of attention—has been estimated to 2.3'' (where 0.8'' can be assumed for the actuation phase), on the basis of Johansson and Rumar’s distribution [12]. The studies are based upon the assessment of the variation of new road risk indexes as a result of the reduced $t_{p,r,a}$; specifically, it is assumed that the I2V communications are such to send on board indications which can supply two levels of alert, namely: the former can take the attention of the driver back to normal levels ($t_{p,r,a}$ equal to 1.46s: green arrow in Fig. 17.2.) and the latter can generate an actual alert ($t_{p,r,a}$ equal to 1.1 s: red arrow).

17.4 The Dilemma Zone and Role of Integrated I2V-ADAS

It is worth reminding the concept of dilemma zone, which has been properly described and analysed in [16]. The so-called dilemma zone is the portion of approach to the intersection, when the traffic light turns into yellow, where the driver is not able to either stop in safety conditions before the stop line (or close to it) or to fully clear the intersection at the end of the yellow light or when the red one is triggered. Such conditions are critical and generate a potential dilemma to the driver, who does not know what his/her behaviour should be in order to act safely, not to commit infractions or cause accidents. Such area can be eliminated with a proper yellow time calculation and if vehicle speed is lower than the established limit, but sometimes it exists and its position and length vary depending upon the cases, therefore some parameters need to be taken into consideration.

In order to clarify the concept of dilemma zone, the behaviour of a driver is considered independently, i.e. irrespectively on the one adopted by the drivers of any vehicles which precede his/her own one. When the yellow light is triggered, the driver is faced with a choice: should he/she stop the vehicle or cross the intersection—even by accelerating—so that he/she can clear the area before the red light? We should keep in mind that—usually—the driver does not know how long the yellow light will last or the so-called clearance time, i.e. the all red time. The solution depends on factors which characterise the distance and time required to stop the vehicle and/or clear the intersection: the initial speed of the vehicle, its features, the actual or possible deceleration, the driver's perception and reaction time, the distance between the stop line and the access, the position of the vehicle when the yellow light is triggered and the extension of the intersection and related geometry. It is obvious that—as a tendency—the drivers who are far from the intersection choose to stop; those who are very close to it—instead—normally try to clear the intersection and therefore—if required and possible—they accelerate.

In either case, the characteristics of the manoeuvres are influenced by the perception—reaction—actuation time ($t_{p,r,a}$) of the driver.

The stopping distance (X_s) is the minimum level of the distance, calculated from the stop line, a vehicle should be within in order to have a comfortable stop and in full safety conditions; beyond such position, the vehicle cannot be stopped (Fig. 17.3). By steady and smooth deceleration, the stopping space can be calculated through a known expression (17.1).

$$X_s = v \cdot t_{p,r,a} + v^2/2a \quad (17.1)$$

where:

- X_s is the *stopping distance* or stopping space of the vehicle [m];
- v is the initial speed of the vehicle [m/s];
- $t_{p,r,a}$ is the perception—reaction—actuation time [s];
- a is the deceleration [m/s^2].

The *clearance distance* (X_c) is the maximum distance from the stop line which identifies a zone where a vehicle can be in order to clear the intersection in full safety

conditions (Fig. 17.3) within a given yellow light time, usually *unknown* by the driver. This was computed through the following formula (17.2), where the acceleration is an optional term, that means equal to zero in case of a constant speed to cross the intersection.

$$X_c = v \cdot (Y + R) + (a/2) \cdot (Y + R - t_{p,r,a})^2 - (W + l_v) \quad (17.2)$$

where:

- X_c is the *clearance distance* [m];
- W is the length of the intersection, measured from the stop line of the approach and depends on the manoeuvre to be performed [m];
- l_v is the length of the vehicle [m];
- v is the speed of the vehicles approaching the intersection, [m/s];
- Y is the duration of the yellow light phase (yellow light time) related to the access which is being taken into consideration [s];
- R is the duration of the all red stage (all red time) [s];
- $t_{p,r,a}$ is the perception-reaction-actuation time [s];

a is the value of the acceleration (assumed as constant) adopted to clear the intersection. In default of more accurate data, such as the ones generated by monitoring, the value of this parameter is assumed through Gazis's equation [14], i.e.: $a[m/s^2] = 4.9 - (0.213 \cdot v[m/s])$

Three different conditions can occur on the basis of the relationship between the two distances which have been defined above.

In the first case ($X_s > X_c$), the dilemma zone results from the overlapping of the Cannot Stop and Cannot Go portions. The position and length of such areas vary from case to case.

The second case ($X_s = X_c$) represents an ideal situation: the dilemma and optional zones disappear; a driver which would find him/herself in those conditions could stop the vehicle or clear the intersection comfortably and in full safety conditions, with no doubts at all on the behaviour to be adopted.

In the last case ($X_s < X_c$), an "optional zone" would generate, i.e. a portion of the access lane where the driver of the vehicle in it may select whether to stop comfortably and safely at the stop line or to clear the intersection in safety conditions.

It is worth mentioning that the dilemma zone for an intersection approach depends on the kinematic parameters of the vehicle (i.e. speed, deceleration or acceleration) besides on the yellow light time, calculated for a specific approach of the intersection. The most appropriate strategy to minimise the issue caused by the presence of the dilemma zone consists of determining a yellow or all red time which allows clearing the intersection from the limit position available to stop. Nevertheless, the variability in the conditions of motion, of the drivers and adherence of the carriageway might determine different circumstances than the ones which are defined a priori. These variations can be observed by means of position detection systems located either on-board the vehicle (GPS-with WAAS, as EGNOS) or on the infrastructure (VIP, Inductive Loops, WSN based on magnetometers, pyroelectric infrared sensors, etc.).

The analyses illustrated hereinunder are aimed at providing a real time estimate of the risk of accident for an approach of road intersections regulated by traffic lights. This information can then be used by an ADAS, which supplies the driver with a risk indicator on the instrument panel; such indicator should be able to resort to information which is usually not available to the driver or which—in any case—he/she cannot calculate in real time, namely: the road in front of him/her (navigator instrument panel), the residual time to the triggering of the red light and the clearance time (I2V), the comparison between the driving dynamics and the safe crossing or stopping conditions. This would allow assessing whether or not a situation is hazardous and—if it is—trying to avoid the potential collision by transmitting alert messages to the potentially involved vehicles.

In the analysis of the safety conditions, we have applied risk indexes formulated on the basis of the vehicle position and speed information.

17.5 Driver's Behaviour and Risk Indexes

The study of the risk of the single vehicle approaching the intersection is strictly linked to the study of the dilemma zone and—subsequently—to the distances required to clear the area and stop depending upon the course state adopted by the vehicle. Two specific indicators have therefore been formulated: the former is relevant to the overall clearance of the intersection and latter refers to the complete stop of the vehicle in correspondence to the stop line. Literature proposes various approaches to risk assessment [21], yet those hereafter described have been originated by our proposal, having in mind a simple approach, at least at this level of analysis.

On the grounds of the analyses described, ratios have been formulated to determine—as a result of the identification of the dilemma zone—simple risk indexes on the basis of specific input data.

With reference to a determined time instant (at a given spatial position D), the risk index relevant to the stop manoeuvre (IR_{stop} or $IR1$) is defined by:

$$IR1 = IR_{stop} = D_{stop} / D \quad (17.3)$$

where:

- D_{stop} is the distance—computed from the stop line—the vehicle needs to stop—in full safety conditions—before or in correspondence to the stop line (stopping distance as previously defined);
- D is the distance—measured from the stop line—where the vehicle is at the time taken into consideration.

According to the report we have presented above, a null or almost null risk index represents the fully safe condition ($D \gg D_{stop}$), since the vehicle can stop without

the risk of occupying the intersection, even if partially. Values of $IR_{stop} \geq 1$, on the other hand, detect potentially hazardous conditions ($D \ll D_{stop}$) for safe stopping. Values of IR_{stop} included between 0 and 1 indicate almost totally safe or almost risky conditions, depending on whether they are closer to zero or to one.

Likewise, a risk index has been defined as related to complete intersection clearance manoeuvre ($IR_{clearance}$ or $IR2$):

$$IR2 = IR_{clearance} = D / D_{clearance} \quad (17.4)$$

where:

- D is the distance—measured from the stop line, where the vehicle is at the instant taken into account;
- $D_{clearance}$ is the clearance distance; such distance, which is computed starting from the stop line, ensures the vehicle the complete clearance of the whole intersection, in full safety conditions, during the yellow light stage (relevant to its manoeuvre) or—in case—during the all red stage. For simplicity, the all red stage has been considered equal to zero in our tests: an all red stage is usually present in real cases, even though it is rather limited in time; in case an all red phase was actually present, the conditions would be implicitly safer than our tests.

Values of $IR_{clearance}$ close to zero identify fully safety conditions ($D \ll D_{clearance}$) for the clearance manoeuvre point of view—i.e. where the vehicle can fully clear the intersection by the end of the yellow light stage relevant to its manoeuvre. On the other hand, high values of risk related to clearance which are greater than or equal to one would identify potentially risky situations ($D \gg D_{clearance}$) for the overall clearance of the area in full safety conditions. Values of $IR_{clearance}$ included between 0 and 1 indicate intermediate conditions.

This section of analysis focused on the behaviour—and relevant criticalities—of the different drivers who approach intersections governed by traffic lights at the moment the yellow light is triggered. The analysis of a single vehicle is not aimed at assessing the consequences of the potential accident; it merely intends to evaluate how much a vehicle—depending upon its dynamics and on the driver's behaviour—is exposed to the risk of accidents: it is a kind of assessment of the exposure to the risk, rather than an estimate of the risk itself.

A numerical tool has been built for such study so that—after the input of specific data of the case examined—the presence and extensions of the dilemma zone could be assessed (Fig. 17.4), as well as the value of the risk indexes (of not completing the manoeuvres of either stop or complete clearance of the intersection by the end of the yellow light or—in case—all red stages) and if there is the actual risk of accident. The tool reproduces the motion of a single vehicle approaching a traffic signal and provides also graphic outputs for the variation of the risk indexes as a function of the initial speeds which can be assumed for the vehicle in exam.

With reference to the three $t_{p,r,a}$ values which have been taken into consideration in the analyses (i.e. 2.3–1.46–1.1 s), the presence and variation of the dilemma area

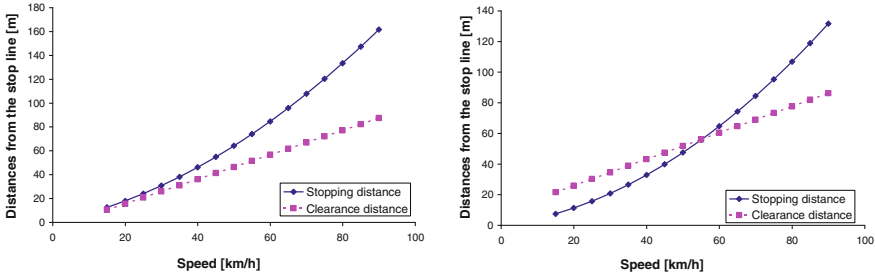
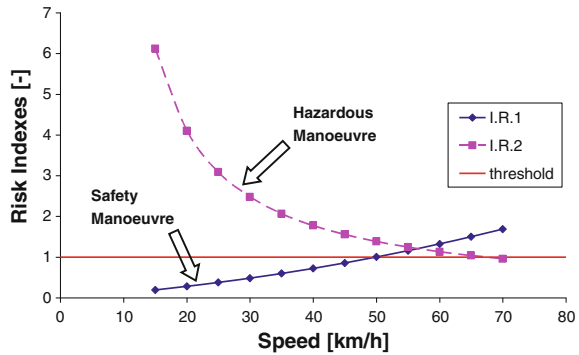


Fig. 17.4 Dilemma zone presence versus vehicle speed in the cases of $t_{p,r,a} = 2.3$ s (left) and 1.1 s (right)

Fig. 17.5 Trend of the risk indexes relevant to the clearance and stop manoeuvres related to a given section and to a specific $t_{p,r,a}$ as a function of the speed

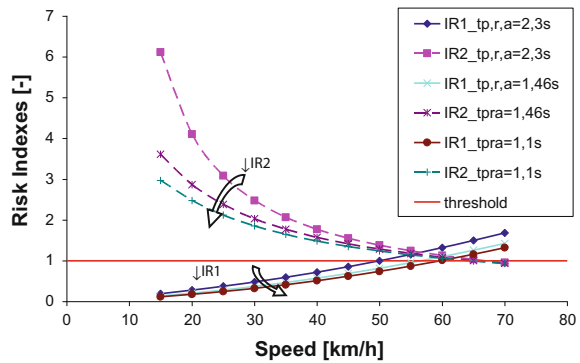


have been investigated to reach the definition of risk indexes relevant to both the clearance and stop manoeuvres; such indexes highlight what the most advantageous or least disadvantageous manoeuvres would be for the drivers of the analyzed vehicle (see the analysis of a specific situation in Fig. 17.5).

In order to detect the risk of the vehicle when approaching the traffic signal, we can assume to update its risk level at different positions before the stop bar, keeping always in mind that the driver has to compare the extension of the crossroad with the stopping distance from his position when he makes the decision either to cross the intersection or to stop. Since the feasible deceleration rates for vehicles fall usually in a quite limited range (a typical range might be between 3 and 5 m/s^2), progressive sections along the approaching lanes can be defined to trace its speed and compare it with the expected value in case of stopping from that distance.

The first of these checking points (named section “A”) is defined assuming a deceleration rate of 3 m/s^2 and is 64 m before the stopping bar, for a vehicle moving with a speed of 50 km/h. Figure 17.5 shows—as related to this specific section (section “A”)—how IR_stop (IR1) grows linearly with the increase of the speed even though the other conditions remain the same, whilst IR_clearance (IR2) decreases in an almost exponential trend.

Fig. 17.6 Comparison diagram of the risk indexes as a function of the speed, relevant to a given section, according to the three $t_{p,r,a}$ which have been taken into consideration



It is worth noticing that the portions of curves above the threshold of $IR = 1$ (which is displayed in red in the diagrams) identify risky situations. If—at a given speed—at least one of the two IR were below such threshold, the manoeuvre to be recommended would be the one which corresponds to it (by a communication on board the vehicle).

If both the IR’s were below such thresholds, then either manoeuvre would not be severely risky and—in any case—it would be appropriate to provide indications on board to apply the safest one, i.e. the one which is farther from the threshold. In those cases where—at a given speed—both indexes were exceeding the $IR = 1$ threshold, then—even though the safety conditions are lacking—it would be appropriate to provide communications on board to apply only the manoeuvre that—between the two ones—would involve lower risk (i.e. the one which is closer to the threshold) or to communicate the risk condition to the other vehicles which are approaching the intersection.

Although here only the risk indexes related to section “A” have been reported, an ADAS can easily update this simple estimation, while the vehicle is approaching the intersection and recognise critical cases by following the evolution of these risk indexes over space/time.

Two typologies of use of the information on the risk indexes above can then be identified. In particular, if the data can be detected in real time, the driver could be informed onboard of a potentially hazardous situation (which might occur if he/she keep such driving behaviour) using—if required and with this purpose—allocated algorithms to predict the trend of the vehicle on the basis of the data detected from the monitoring; another use, which is strictly linked to the utilisation of the diagrams obtained, would be detecting—in case the vehicle were already within the dilemma zone—the lowest risk manoeuvre and sending a message on board to inform the driver.

Furthermore, the effects of using communication technologies between the infrastructure and the vehicle have been assessed reducing—in the analysed situations—the driver’s $t_{p,r,a}$ from 2.3 to 1.46 and 1.1 s, leaving the other conditions unchanged.

Figure 17.6 reports an example of a diagram which summarises the curves of the risk indexes assessed for the three different $t_{p,r,a}$. It is worth noticing how—as a result of the reduced $t_{p,r,a}$ —the risk indexes relevant to both the clearance and stop manoeuvres result to be reduced as well.

17.6 Conclusions

In this work an ADAS for traffic signal approaching has been analysed and two main roles have been considered:

- provide a risk estimation for *alternative manoeuvres* (stopping or clearance) and then communicate the driver the less hazardous manoeuvre on the basis of known, measured or estimated parameters;
- reduce the risk level, by reducing the driver perception and reaction time, since I2V/IVC can increase the level of attention of the driver.

The experiments run in simulation by means of a spreadsheet have led to acknowledge—as a result of the reduction in the $t_{p,r,a}$,—a corresponding reduction in the estimated risk of accidents. The positive effects of the infrastructure-vehicle communication have been ascertained in terms of reduced exposure to the risk by a single vehicle (analyses of the trend of the single vehicle approaching the intersection regulated by traffic lights). More specifically, as related to the behaviour of a driver at the moment the yellow light is triggered for his/her traffic stream, the application of I2V systems (corresponding to a reduction in the $t_{p,r,a}$), the following effects have been observed:

- a reduction of the dilemma zone extension;
- the disappearance of the dilemma zone and growth of the zone of choice: in some cases, as a result of the increased level of attention in order to attain standard values, i.e. $t_{p,r,a}$ equal to 1.46 s and—in a large number of cases relevant to the forwarding of alert messages— $t_{p,r,a}$ equal to 1.1 s;
- the decreasing of the risk indexes referred to the stop (IR1 o IR_stop) and clearance (IR2 o IR_clearance) manoeuvres, mainly at the speed values corresponding to IR which were far greater than the safety threshold (IR = 1);
- the advanced knowledge of IR1 and IR2, with the subsequent opportunity to warn the drivers on board (possibly before they enter the dilemma zone) about the lowest risk manoeuvre to be undertaken;
- the opportunity to reduce instantaneously, and therefore in real time, the risk or—better—the exposure to the risk—of not completing in full safety conditions the manoeuvre which is intended to be undertaken by the end of the yellow light stage.

In short, the results of the analyses show that the use of the I2V e V2I communication systems in the intersections regulated by traffic lights—assumed in the processing

as directly related to a reduction of the $t_{p,r,a}$ —has beneficial effects on road safety as related to the reduction of risks of accidents.

We have also intended to allocate the all red as safeguard fraction for those drivers whose behaviour—perhaps because of slower reflexes—is not within the average one which was computed in these analyses; an advanced ADAS system may include the *transmission on board of the all red time*, consequently modifying the risk conditions.

Furthermore, the analyses performed allow supporting also the combined use of sensors, to enable the most viable continuous monitoring and assess the dilemma zone and the potential risk of accident instantaneously and in real time.

It is also worth specifying that the analyses did not consider any actual data on the use of the I2V technologies—since they are not available to date—or any active intervention on the vehicle in case of need.

The presented risk index is related to the change of phase of the traffic lights and it can be used with other risk indexes, based on the conflict analysis, as those of the frontal, rear end or crossing collision, which are already related to the vehicle trajectories output of traffic micro-simulation models.

To clear or cross a crossroad could be a suggestion suitable for those vehicles which are closer to the stopping line, but it should not generate confusing cases for drivers, such as suggesting to cross to a following vehicle, while suggesting the leader to stop.

The subject is in evolution and many questions remain open. For example, an investigation of vehicle behaviour, when it is not isolated in approaching the traffic signal, need more tests, possibly also with a traffic micro-simulation tool. It can be assumed that the actual potential of the systems which have been taken into consideration could be assessed once said technologies are widely spread on the market.

References

1. Archer, J.: Indicators for traffic safety assessment and prediction and their application in micro—simulation modelling: a study of urban and suburban intersections. Doctoral Thesis, Royal Institute of Technology, Stockholm, Sweden 2005
2. Benz, T., Gaitanidou, E., Spyropoulou, I., Yannis, G., Tapani, A.: Modelling road traffic safety—The in-safety approach. In: Proceedings of the 13th World Congress and Exhibition on Intelligent Transport Systems and Services, ERTICO, London (2006)
3. Coopers project co-operative systems for intelligent road safety. <http://www.coopers-ip.eu/index.php?id=project>. Accessed 07 Mar 2012
4. CoVeL—Cooperative Vehicle Localization for Efficient UrbanMmobility. <http://www.covel-project.eu/>. Accessed 07 Mar 2012
5. CVIS Cooperative Vehicle-Infrastructure Systems. <http://www.cvisproject.org/>. Accessed 07 Mar 2012
6. Dalla Chiara, B., Deflorio, F.P., Diwan, S.: Assessing the effects of inter-vehicle communication systems on road safety. IET Intell. Transp. Syst. **3**(2), 225–235 (2009). doi:10.1049/iet-its:20080059

7. Day, C.M., Premachandra, H., Brennan, T.M., Sturdevant, J.R., Bullock, D.M.: Operational evaluation of wireless magnetometer vehicle detectors at signalized intersection, transportation research Record 2192
8. FOTsis—European Field Operational Test on Safe, Intelligent and Sustainable Road Operation. <http://www.fotsis.com/>. Accessed 07 Mar 2012
9. Gettman, D., Pu, L.: Theoretical validation of surrogate safety assessment methodology for roundabouts and cross intersections. In: Proceedings of the 13th World Congress and Exhibition on Intelligent Transport Systems and Services, London (2006)
10. Hurwitz, D. S.: Application of driver behavior and comprehension to dilemma zone definition and evaluation. Open Access Dissertations, Paper 112, University of Massachusetts (2009)
11. INTERSAFE-2—cooperative intersection safety. <http://www.intersafe-2.eu/public/>. Accessed 07 Mar 2012
12. Johansson, G., Rumar, K.: Driver's brake reaction times. *Hum. Factors* **13**(1), 23–27 (1971)
13. Ki-Joon, K., Jaehoon, S.: Development of intersection traffic accident risk assessment model—application of micro-simulation model with SSAM to Sungnam City. In: IRTAD Conference. Seoul, Korea (2009)
14. Klein, L.A., Mills, M.K., Gibson, D.R.P.: Traffic Detector Handbook, 3rd edn, Chapter 4. Federal Highway Administration. Report No. FHWA-HRT-06-139. <http://www.fhwa.dot.gov>. Accessed 07 Mar 2012 2006
15. Klunder, G., Abdoelbasier, A., Immers, B.: Development of a micro-simulation model to predict road traffic safety on intersections with surrogate safety measures. In: Proceedings of the 13th World Congress and Exhibition on Intelligent Transport Systems and Services, London (2006)
16. Liu, C., Herman, R., Gazis, D.C.: A review of the yellow interval dilemma. *Transp. Res. Part A: Policy Pract.* **30**(5), 333–348 (1996)
17. MacNeill, P., Miller, R.: A new technology for a cruise control system a new technology for a cruise control system. IEEE Vehicle Technology Conference (2003)
18. Maile, M., Delgrossi, L.: Cooperative intersection collision avoidance system for violations (CICAS-V) for avoidance of violation-based intersection crashes. *Enhanced Safety of Vehicles 2009*
19. Monteil, J., Billot, R., El Faouzi, N.E.: Towards cooperative traffic management: methodological issues and perspectives. In: Proceedings of Australasian Transport Research Forum, pp. 28–30, Adelaide, Australia, September 2011
20. Morsink, P.L.J., Wisman, L.J.J., Dijkstra, A.: Micro-simulation for road safety impact assessment of advanced driver assistance systems. In: European ITS Congress, Geneva (2008)
21. Rausand, M.: Risk assessment, theory, methods, and applications. Norwegian University of Science and Technology, Trondheim. ISBN: 978-0-470-63764-7, (2011)
22. SAFESPOT integrated project. <http://www.safespot-eu.org/>. Accessed 07 Mar 2012
23. Tapani, A.: Traffic simulation modelling of driver assistance systems. In: 16th world congress on ITS, (2009)
24. Torday, A., Baumann, D., Dumont, A.G.: Road safety assessment using micro simulation based indicators. In: Proceedings of the ITS World Congress, Madrid (2003)
25. US Department of Transportation: Federal Highway Administration. Surrogate Safety Assessment Model (SSAM), Washington (2009)
26. US Department of Transportation: Cooperative Intersection Collision Avoidance System Limited to Stop Sign and Traffic Signal Violations (CICAS-V), Task 8 Final Report, Prototype Build and Testing, (Appendix F), June 24, August 29, 2008

Part III
Signal Processing, Sensors, Systems
Modeling and Control

Chapter 18

A Component-Oriented Model for Wastewater Pumping Plants

Mohamed Abdelati, Felix Felgner and Georg Frey

Abstract A typical wastewater pumping plant comprises a screening process and a pumping process. The first process separates coarse material out of wastewater, while the second one boosts the wastewater toward a treatment facility. Appropriate component models for such plants are hardly found in literature. Indeed, there exist standard component models in all-purpose fluid simulation tool libraries; their generality, however, makes those models too complex to be used for wastewater pumping plants. The lack of models forces engineers to test their control scenarios on real implemented systems, which may lead to unexpected delays and painful costs. In this work, easily manageable component-oriented models are derived and applied to the modeling and simulation of a real wastewater pumping system. The model derived here is implemented in the component-oriented Modelica language, and it helps better understand the system dynamics. Thereby, a tool is provided for evaluating the performance of possible control schemes.

Keywords Modelica · Wastewater · Modeling · Simulation · Automation.

18.1 Introduction

Daily amounts of about 15,000 m³ of partially treated wastewater are pumped through the new terminal pumping station (NTPS) located at northern Gaza to the new wastewater treatment plant (WWTP). Once the construction of this new treatment plant

M. Abdelati (✉)

Electrical Engineering Department, IUG, Gaza, Palestinian Territories

e-mail: muhammet@iugaza.edu

F. Felgner · G. Frey

Chair of Automation, Saarland University, Saarbrücken, Germany

e-mail: felix.felgner@aut.uni-saarland.de

G. Frey

e-mail: georg.frey@aut.uni-saarland.de

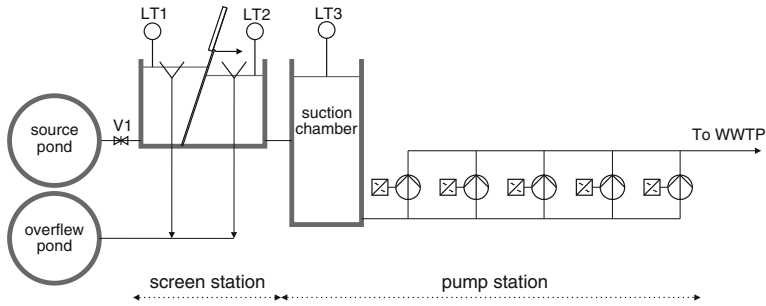


Fig. 18.1 New terminal pumping station (NTPS)

is completed, the pumping rate will reach an average of 35,000 m³/day. The transmission pipe has 7.6 km length, 80 cm diameter and 26 m static head. At the present phase a group of ponds near the pumping station are used to buffer and partially treat the wastewater collected from northern Gaza [1]. Operators of the pumping station manually control the intake amount of these ponds and allow it to be pumped to the wastewater treatment plant. The manual operation should be replaced by an automation system. To this end, models for evaluating different control schemes are necessary. These models should allow efficient simulation of the overall system over long time horizons, to validate the system behavior especially under abnormal conditions like e.g. power outages which are quite common in Gaza.

The pumping station is equipped with 5 booster pumps from ABS. Each pump has a power rating of 315 kW and an expected head of 38 m. It has a pumping capacity of 360 kg/s while running safely at a maximum speed of 1,300 rpm. The suction chamber of the booster pumps has a capacity of 500 m³ and equipped with a level transmitter which is used to control the operation of the booster pumps [2]. In [3], a model of the wastewater recovery system was developed. The work presented here continues the project by presenting a component-oriented model of the wastewater pumping plant. The control scheme of the pumping process will be detailed in Sect. 18.2 and modeled in Sect. 18.3. The simulation results will be presented in Sect. 18.4 and finally, in Sect. 18.5, we will give concluding remarks and outline the future application of the model.

18.2 Functional Description

The new terminal pumping station (cf. process flow diagram in Fig. 18.1) transports wastewater from the northern part of Gaza city to the new wastewater treatment plant for Northern Gaza. It basically consists of a screen station and a pump station [4].

18.2.1 Screen Station

The screen chamber receives the wastewater from the source pond by gravity force. The manual valve (V1) allows setting the intake flow rate. The screen separates coarse material out of wastewater. The coarse material is loaded into a conveyor system. The rack screen and the conveyor starts at a signal due to difference in levels of level transmitters (LT1) and (LT2) located in front of and behind the screen respectively. They run during a preset time to leave at pause position. If the outtake of the screen fails to compensate the intake, wastewater starts to accumulate in the screen chamber. Overflow occurs if accumulation reaches a specific level (1.6 m). The overflow is collected in a dedicated pond where it is recharged to the plant by a minor process that is not addressed in this chapter.

18.2.2 Pump Station

This station pumps the wastewater from the suction chamber to the new wastewater treatment plant. The bottom of the suction chamber is placed 2.3 m below the bottom of the screen chamber. The booster pumps are controlled and operated by a signal from level transmitter (LT3). Due to efficiency concerns, pumps are not allowed to run at speeds below one third their nominal speeds. The first pump in operation starts at level L6. When one pump is in operation, a flow of about 120–360 kg/s (corresponding to 33–100 % rotational speed) will be pumped using the frequency converter to keep a preset value of the level (L5) in the suction chamber. The first pump stops at level L1.

If the first pump operates at 100 % capacity and the level increases, the second pump starts at level L7. When two pumps are in operation, a flow of about 360–720 kg/s (corresponding to 50–100 % rotational speed) will be pumped using the frequency converters to keep a preset value of the level (L5) in the suction chamber. Each pump in operation will pump 180–360 kg/s equal. The second pump stops at level L2.

If two pumps operate at 100 % and the level increases up to level L8, the third pump starts. When three pumps are in operation, a flow of about 720–1,000 kg/s (corresponding to 66–100 % rotational speeds) will be pumped using the frequency converters to keep a preset value of the level (L5) in the suction chamber. Each pump in operation will pump 240–333 kg/s. The third pump stops at level L3.

If three pumps operate at 100 % and the level increases up to level L9 the forth pump starts. When four pumps are in operation, a flow of about 1,000–1,200 kg/s (corresponding to 75–100 % rotational speeds) will be pumped using the frequency converters to keep a preset value of the level (L5) in the suction chamber. Each pump in operation will pump 250–300 kg/s. The forth pump stops at level L4. It should be noted that the previous flow rates are predicted values. Actual quantities depend on the resulting dynamic head which is almost proportional to the square value of the

Table 18.1 Level threshold settings

Level No.	Activity	Setting (m)
L9	Start level P4	2.3
L8	Start level P3	2.1
L7	Start level P2	1.9
L6	Start level P1	1.8
L5	Set point level	1.8
L4	Stop level P4	1.7
L3	Stop level P3	1.6
L2	Stop level P2	1.4
L1	Stop level P1	1.2

flow rate as will be highlighted in the simulations. A maximum of four pumps can be concurrently operated leaving the fifth one as standby unit. Typical values for the level setting are summarized in Table 18.1.

18.3 Model Derivation

For the modeling of large fluid systems, standard component models as found in simulation tool libraries [5] are very complex [6]. The generic concept behind these models makes them widely applicable for different systems. However, this also leads to a complex parameterization and large simulation overhead. In the presented project, it is not intended to build a sophisticated model for detailed investigations rather to conclude with a manageable working model. The desired simulation model is required to provide better understanding of the pumping process dynamics. Moreover, it is intended to be a tool for testing and improving proposed control schemes. To this end, we used the modeling and simulation environment Dymola which is based on the component-oriented modeling language Modelica [7].

In [3] a water recovery system has been modeled using the same approach. The system included source/sink, pipes, pumps and other components. The water network components are interconnected through a liquid connector, where conservation of mass flow is assumed. The pressure of the liquid (wastewater in our case) at the connector is referred by p and the mass flow rate is referred by q . In the following subsections the new components necessary to build the system model will be derived.

18.3.1 Screening Process

The screening process may be decomposed to the following components: two buffering chambers (one after the inlet and one before the outlet), a Bar screen crossing between the buffer chambers, and a screen controller. This is illustrated in Fig. 18.2.

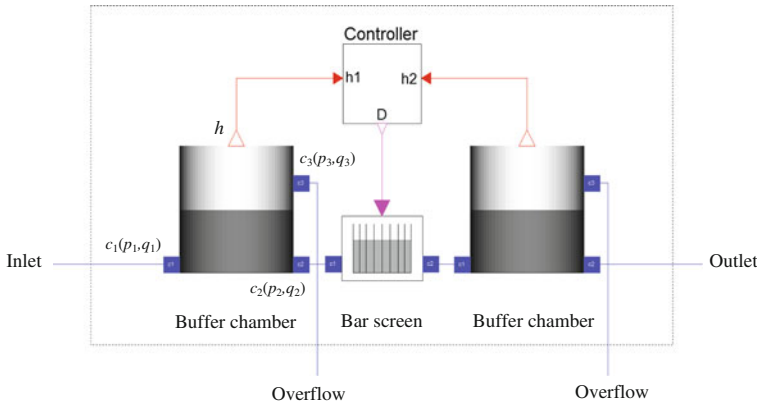


Fig. 18.2 Screening process

The buffer chamber model has two water connectors at its base and a third one located at the overflow level (h_{of}) measured from the chamber’s base. Another connector of type real output is added to deliver the liquid height (h) to the screen controller.

The pressure at the base connectors c_1 and c_2 is given by:

$$p_1 = p_2 = \rho gh \tag{18.1}$$

where ρ is the wastewater density, g is the acceleration due to gravity, and h is the wastewater level in the chamber. The pressure at the overflow connector c_3 is discontinuous at the threshold height h_{of} (equals 1.6 m in our case) as follows:

$$p_3 = \begin{cases} 0, & h \leq h_{of} \\ \rho g(h - h_{of}), & h > h_{of} \end{cases} \tag{18.2}$$

The wastewater level is related to the mass flow rate in the ports as follows:

$$\frac{dh}{dt} = \frac{q_1 + q_2 + q_3}{\rho A} \tag{18.3}$$

where A is the cross sectional area of the chamber.

The bar screen is the interface located between the two buffering chambers. The wastewater flow across this section may be visualized as shown in Fig. 18.3.

At a given time, the screen collects an amount of coarse material causing a friction resistance to the water flow. This results in a difference in the wastewater levels across the screen. The waste water flow has two components; the first one (q_a) for which the flow crosses the screen facing atmospheric pressure (from h_2 to h_1). The other remaining component is referred by (q_b) as illustrated in Fig. 18.4.

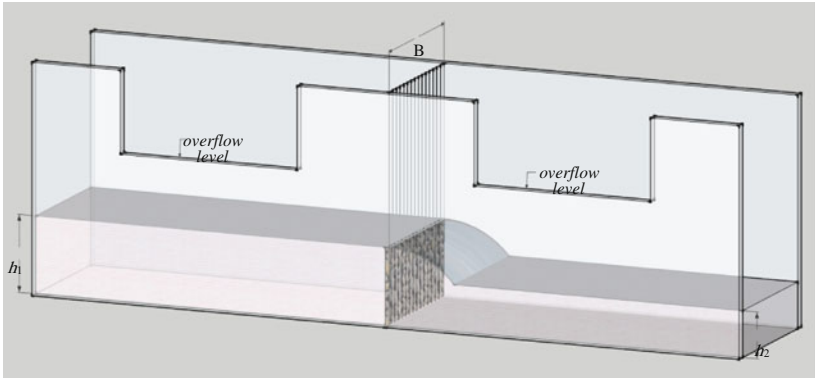
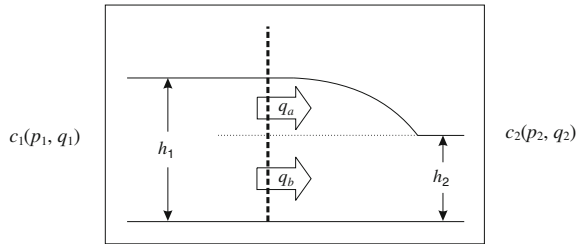


Fig. 18.3 Bar screen visualization

Fig. 18.4 Decomposing the flow across the screen



The values of these components may be derived from the Bernoulli equation. Under the assumption that the screen area is much less than the area of the inlet buffer base, the following results are obtained:

$$q_a = \frac{2}{3} G B \rho \sqrt{2g} (h_1 - h_2)^{3/2} \tag{18.4}$$

$$q_b = G B \rho \sqrt{2g} h_2 (h_1 - h_2)^{1/2} \tag{18.5}$$

where B is the screen width and G is a transparency coefficient that represents the screen conductivity ranging between one and zero. At a given time instant it is related to the amount of the coarse material accumulated on the screen. Therefore, it is a function of the total mass flow (Q) across the screen and a function of the wastewater quality. In analogy with charging a capacitor, we model G as

$$G = e^{-Q/Q_0} \tag{18.6}$$

where Q_0 is a factor that reflects the wastewater quality. In simulations, we treated it as a constant equal to 8,64,000, forcing G to be about 1% after working for 5 h at the maximum mass flow rate.

Fig. 18.5 Bar screen icon

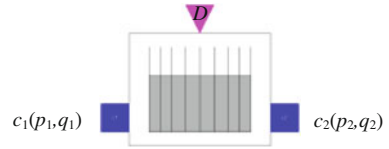
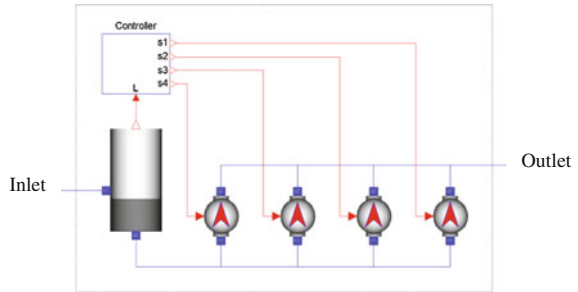


Fig. 18.6 Pumping process



The total mass flow since $t = t_0$ and the connectors' signals are related as follows:

$$p_1 = \rho gh_1 \tag{18.7}$$

$$p_2 = \rho gh_2 \tag{18.8}$$

$$q_1 = q_2 = q_a + q_b \tag{18.9}$$

$$\frac{d}{dt} Q = q_1; \quad Q(t_0) = 0 \tag{18.10}$$

where $Q(t_0)$ is the initial value. When the screen is triggered by a digital signal (D) to discharge accumulated coarse material, at $t = t_0$, Q is re-initialized by 0. The bar screen module is graphically represented as illustrated in Fig. 18.5.

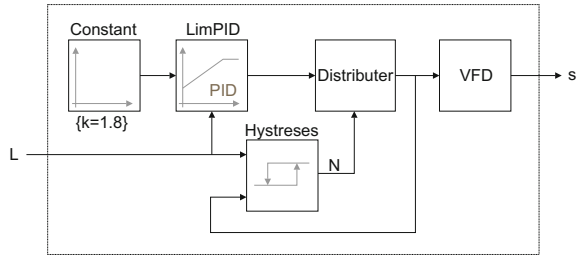
The screen controller senses the amount of accumulated coarse material by means of the wastewater levels in the buffer chambers. The controller activates the discharge signal (D) whenever $h_1 - h_2 > h_t$ where h_t is a preset value taken 20 cm in the simulations.

18.3.2 Pumping Process

This process contains a suction chamber, booster pumps and a controller as illustrated in Fig. 18.6. The suction chamber model is governed by the following equations:

$$p_1 = \begin{cases} 0, & L \leq 2.3 \\ \rho g(L - 2.3), & L > 2.3 \end{cases} \tag{18.11}$$

Fig. 18.7 Pumping process controller



$$p_2 = \rho g L \tag{18.12}$$

$$\frac{dL}{dt} = \frac{q_1 + q_2}{\rho A} \tag{18.13}$$

The level signal (L) is measured relative to the chamber’s bottom, which is located 2.3 m below the inlet connector. A liner head-flow characteristic around the nominal operating point is used for the booster pumps [3] as follows:

$$q_1 = \frac{s}{s_n} \left[q_n - a \left(\frac{p_2 - p_1}{\rho g} - h_n \right) \right] \tag{18.14}$$

where a is the additive inverse of the slope of the flow-versus-head curve at the nominal operating point (h_n, q_n), is the respective pump’s speed, and s_n is the nominal speed. The specific data for the installed pumps are: $a = 8.3 \text{ kg/s/m}$, $h_n = 38 \text{ m}$, $q_n = 360 \text{ kg/s}$, and $s_n = 1, 300 \text{ rpm}$.

In order to implement the control scheme specified in Sect. 18.2.2, the calculation of the speed vector, $\mathbf{s} = (s_1, s_2, s_3, s_4)^T$, according to the level signal (L) is decomposed as shown in Fig. 18.7. The controller has a PID module with limited output, anti-windup compensation and set point weighting [8]. Its output specifies the required pumping capacity, which has a minimum of 0 when all pumps are off and a maximum of $4 \times 1, 300$ when 4 booster pumps run at their full speed. The hysteresis module decides on enabling or disabling each pump and calculates the number of enabled pumps (N). The implementation of this module will be described later. The distributer divides the PID output value by this number and assigns the result to the enabled pumps taking into account that the result has a saturation value of 1,300rpm. The hysteresis module is a sequential circuit which uses the state of pumps (enabled/disabled), their assigned speed, and the wastewater level to calculate their next states. Then it calculates the number of enabled pumps and delivers it to the distributer module. The state equation of a pump is simply the characteristic equation of an RS flip flop which is set whenever wastewater level exceeds the set level of the pump and reset whenever the level drops below the stop level or its speed drops below its minimum allowed speed. The minimum speed limits of pumps are 433, 650, 866, and 975 rpm, respectively. This ensures exempting a pump whose

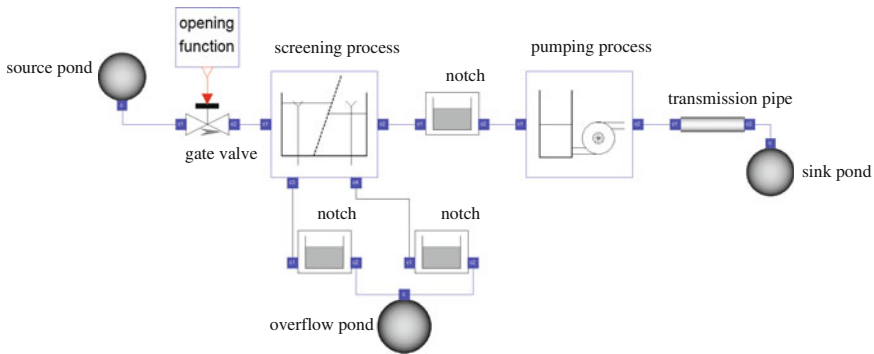


Fig. 18.8 System model

load share can be carried by the other running pumps. The Variable Frequency Drive (VFD) is modeled by a first-order block with a time constant of 5 s resulting in an acceleration time of about half a minute to move forward or backward between zero speed and rated speed states.

18.3.3 Complementary Modules

Encapsulating the screening and pumping processes into two stand alone modules, the system model will be as illustrated in Fig. 18.8. Connectors of the overflow pond and the sink pond are located above the surfaces of the ponds. Therefore, they have the atmospheric pressure value which is our reference ($p = 0$). On the other hand, the connector of the source pond is located at the bottom of the pond. The wastewater level in the source pond may vary along the year depending on the collected sewage. However, the pond is so huge that its level is safely considered as constant on weekly or even monthly bases. Consequently, the pressure at the source pond connector is treated as constant. This constant is taken as the value found during the month of June which is about 0.25 bar.

The gate valve is adjusted manually to control the daily transmitted wastewater and indirectly decide on the pumping capacity. If the inlet wastewater rate exceeds the feasible pumping capacity, then the automation system should signal a high level alarm prior to overflow. The operator in turn, should react immediately by decreasing the gate valve opening and vice versa. Operator interaction is expected to be on weakly bases in case pumping is done all the day. A linear relation between flow and pressure drop is used for the valve model. The control signal of the valve is named *opening* and its value ranges from 0 at full closure to 1 at full opening. The nominal hydraulic conductance of a valve, k , is defined as the ratio of nominal flow to nominal pressure drop at full opening. Assuming linear pressure drop, then the flow is governed by the following equation:

$$q_1 = \text{opening} \times k \times (p_1 - p_2) \quad (18.15)$$

The Bernoulli equation is used to derive the model of notches. Since they always have inlet pressure greater than or equal to the corresponding outlet pressure, their equation reduces to

$$q_1 = \frac{B}{g} \sqrt{\frac{2}{\rho}} \left[\frac{2}{3} (p_1 - p_2)^{3/2} + p_2 (p_1 - p_2)^{1/2} \right] \quad (18.16)$$

where B is the width of the notch.

The transmission pipe is modeled according to the Hazen-Williams equation [9]. The resulting model is

$$p_1 - p_2 = \frac{10.67gl}{C^{1.85} \rho^{0.85} d^{4.87}} q_1^{1.85} + \rho g H \quad (18.17)$$

where d is the diameter, l is the length, H is the static head, and C is the roughness coefficient of the pipe. This coefficient is about 140 for most pipes as it does not depend so much on the roughness of the material itself, but on the roughness of the bacterial slime layer which grows on the pipe wall.

18.3.4 The Implementation Procedure

This section describes briefly how the models were implemented in Modelica using the Dymola tool [10]. The first step was implementing the liquid connector (c). Its icon is represented by a small blue square and it is defined as follows:

```
connector c
  Modelica.SIunits.Pressure p;
  flow Modelica.SIunits.MassFlowRate q;
end c;
```

Then, the components necessary to build the top level module are implemented one by one. For example, the suction chamber has two liquid connectors (c_1 and c_2) in addition to an output connector (L) of type real. Being governed by Eqs. 18.11, 18.12, and 18.13, its Modelica code will be as shown in Fig. 18.9. Only the equations section along with the necessary parameters and constants need to be written by the programmer. The instantiations of connectors are simply done by dragging-and-dropping in the graphical interface of Dymola. Moreover, the tool allows drawing a graphical icon to represent the component. It also generates code for graphically interconnected components that build a higher level module. For example, Fig. 18.10. illustrates the Modelica code that corresponds to the screening process shown in Fig. 18.2.

Fig. 18.9 Modelica code of the suction chamber

```

model SuctionChamber
  constant Real g=Modelica.Constants.g_n;
  parameter Modelica.SIunits.Density rho=1200 "liquid density";
  parameter Modelica.SIunits.Area A=125 "base area";
  parameter Modelica.SIunits.Length L_0=0 "initial height";
  Modelica.SIunits.Length L(start=L_0,min=0) "hight";
  Connectors.c c1 "inlet";
  Connectors.c c2 "outlet";
  Connectors.OutPort Level;

equation
  c1.p =if L>2.3 then rho*g*(L-2.3) else 0;
  c2.p = rho*g*L;
  der(L)=(c1.q+c2.q)/rho/A;
  Level.signal[1]=L;
end SuctionChamber;

connector MyPackage.Components.Connectors.c
  Modelica.SIunits.Pressure p;
  flow Modelica.SIunits.MassFlowRate q;
end c;

connector MyPackage.Components.Connectors.OutPort
  "Connector with output signals of type Real"
  parameter Integer n=1 "Dimension of signal vector";
  replaceable type SignalType = Real "type of signal";
  output SignalType signal[n] "Real output signals";
end OutPort;

```

Fig. 18.10 Code generated by the graphical tool

```

model ScreeningProcess
  Connectors.c inlet;
  Connectors.c outlet;
  Connectors.c overflow1;
  Connectors.c overflow2;
  BufferChamber bufferChamber1;
  BufferChamber bufferChamber2;
  BarScreen barScreen;
  ScreenController screenController;
equation
  connect(inlet, bufferChamber1.c1);
  connect(bufferChamber1.c2, barScreen.c1);
  connect(barScreen.c2, bufferChamber2.c1);
  connect(bufferChamber2.c2, outlet);
  connect(bufferChamber1.c3, overflow1);
  connect(bufferChamber2.c3, overflow2);
  connect(bufferChamber1.h, screenController.L1);
  connect(bufferChamber2.h, screenController.L2);
  connect(screenController.Discharge, barScreen.Discharge);
end ScreeningProcess;

```

A good modeling methodology starts by implementing simple models, which can be easily verified by intuition. It continues with models of increasing complexity until reaching the top-level module. At each stage, created components are connected to form system models whose simulation results can be compared to expectations from the *mind* model [11]. If they agree, the model is verified. Otherwise, the mathematical model is revised or the mind model is adjusted through gaining new physical insight.

18.4 Simulation Results

The aim of simulations is to validate the consistency of the derived model and to investigate the behavior of the system under unfavorable scenarios. Power failure, excess flow, and operator inattentiveness may lead to overflow. The implemented

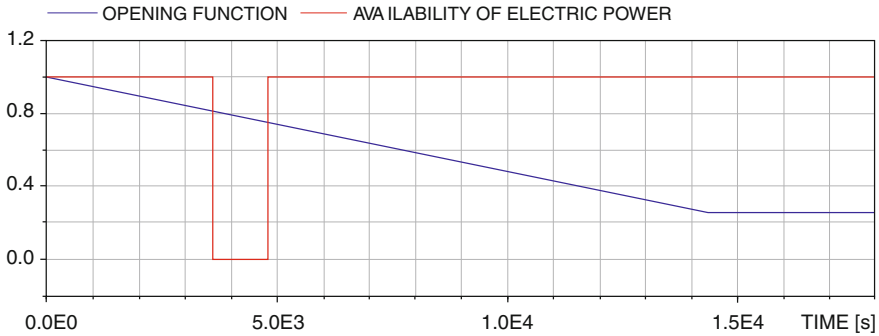


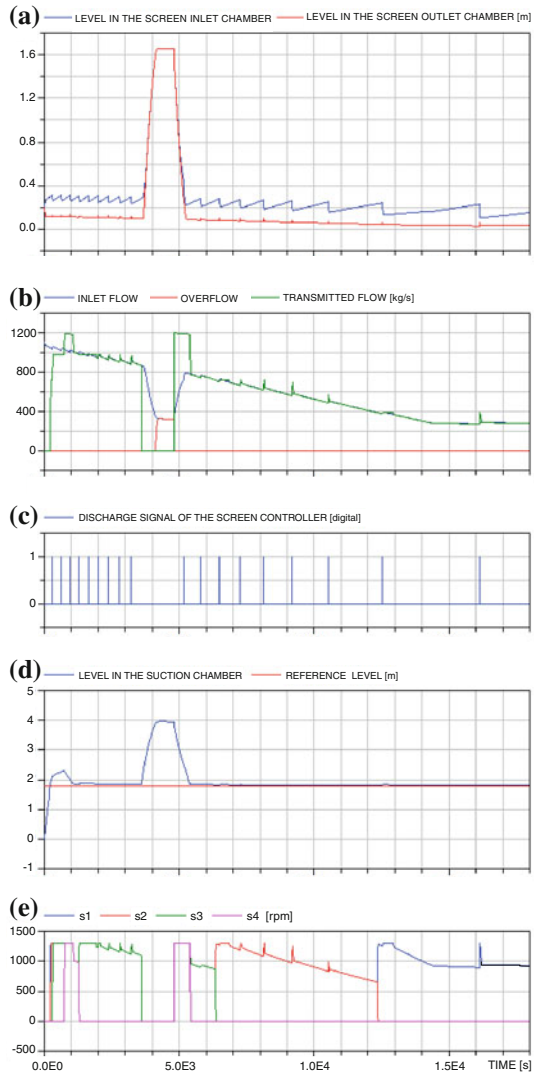
Fig. 18.11 Simulation scenario

control algorithm is widely used in pumping stations and claims to minimize the number of restarts of pumps and the number of running pumps given a desired daily flow. The running period and the number of restarts have direct impact on the depreciation of pumps and their power consumption. Deriving a formula for the cost function, which also includes overflow cost and operator satisfaction (utility), is intended to be done in a future work. The scope of this chapter is to create a working model and simulate normal and unfavorable running conditions leaving the evaluation and improvement of the control algorithm to a complementary work. To this end, the opening signal of the gate valve and the availability of electric power are assigned the functions shown in Fig. 18.11.

The opening signal function represents possible flows along the year. The failure of electric power supply during the time period [3,600 s, 4,800 s] is intended to investigate overflow behavior during a relatively high flow period. The failure is implemented by suppressing the controllers' outputs during the failure period.

Simulation results which lie within our interest are shown in Fig. 18.12. As expected, they are consistent with real world data observed in the plant. In Fig. 18.12a, the waste water levels in the screen chambers are depicted. Shortly after the power failure, the levels in the chambers reach the level of the overflow notches (1.6 m) and eventually cause flood to the overflow pond as shown in Fig. 18.12b. In the same figure, the inlet and transmitted flows are depicted. It is worth to highlight the value of the outlet flow when the pumps run at their full capacity. It is about 380, 720, 982, and 1,186 kg/s while the number of pumps equals 1, 2, 3, and 4, respectively. Figure 18.12c illustrates the time instances when the screen discharge signal is enabled. This happens whenever the level difference in the screen chambers reaches 0.2 m. The wastewater level in the suction chamber is depicted in Fig. 18.12d. Apart from the starting and the power failure periods, the level in the suction chamber is almost equal to the reference value which is 1.8 m. The overshoot is expected as it is necessary to trigger the starting of the pumps. The load share of these pumps is shown in Fig. 18.12e. Running pumps always have equal shares, as desired, and it is notable that sudden increase in the pump speeds occurs at screen cleaning instants.

Fig. 18.12 Simulation output results



Keeping the wastewater level in the suction chamber around the reference value implies adjusting the outlet flow to meet the inlet flow. However, this does not insure running the plant in the most efficient way. A controller is required which optimizes a proper performance measure. In [12], we applied the models to compare two control schemes (cyclic on/off control and variable speed control), while the performance was measured by respecting the loss in the motors' drivers and the number of restarts (which affect the maintenance cost). The simulation showed that cyclic on/off control yields an average energy saving of 167kWh/day (2%).

18.5 Summary and Outlook

This work presents an easily manageable model for a wastewater pumping station in northern Gaza. The resultant model provides a practical tool for examining the system control under different running conditions, such as pump failure and changing flow rates. This simulation model assists in adjusting the control reference points and parameters to cope with regular and undesired situations. The simulated control algorithm is widely used in pumping stations and it is believed that it works to minimize maintenance and running costs by minimizing the number of running pumps and limiting the number of their restarts for a certain inlet rate.

At the present phase, the ponds of the old treatment plant serve as a buffer for the wastewater before being pumped via the NTPS to the new treatment plant. This buffering stage will not be available by the completion of the project as the old plant will be removed and incoming wastewater is planned to be transmitted directly to the new treatment plant. Only one small size pond will be left for collecting emergency overflows at the pumping station. As a result, the real challenge of the control problem is not the present phase where a fixed daily amount of wastewater needed to be transported. In the final phase, the pump station must handle instantaneous variation of the wastewater flow. Accidental overflow will result in an additional repumping cost and undesired environmental consequences. Therefore, an estimate of the daily diurnal flow pattern is necessary to examine the plant controller under daily variation of wastewater flow.

In a future work, we will employ simulations over long time horizons to respect special conditions found in Gaza but also many other developing areas. For example, frequent failure in the main electric power supply is common in Gaza city nowadays and requires intensive operator supervision. Moreover, power produced by the standby generators is much more expensive than the power of the main supply. This is a point normally not considered in deriving the control laws. Models implementing functions to derive the total cost of operation similar to models presented in [13] combined with predictive—simulation-based—hybrid control schemes as in [14] are expected to be of great value under these conditions.

Acknowledgments The authors would like to express their gratitude to Alexander von Humboldt Foundation for supporting this work.

References

1. Werner, M., et al.: North Gaza emergency sewage treatment plant project—environmental assessment study. Technical report, Engineering and Management Consulting Center, Gaza (2006)
2. Abdelati, M., Rabah, F.: A framework for building a SCADA system for Beit Lahia wastewater pumping station. *Islamic Univ. J. (Nat. Stud. Eng. Ser.)* 15(2), 235–245, ISSN 1726–6807 (2007)

3. Abdelati, M., Felgner, F., Frey, G.: Modeling, simulation and control of a water recovery and irrigation system. In: Proceedings of the 8th International Conference on Informatics in Control, Automation and Robotics. Noordwijkerhout, The Netherlands (2011)
4. Palestinian Water Authority. Bidding documents for the construction of terminal pumping station. NGEST Project Management Unit, Gaza (2004)
5. Elmqvist, H., Tummescheit, H., Otter, M.: Object-oriented modeling of thermo-fluid systems. In: Proceedings of the 3rd International Modelica Conference, Linköping (2003)
6. Link, K., Steuer, H., Butterlin, A.: Deficiencies of Modelica and its simulation environments for large fluid systems. In: Proceedings 7th Modelica Conference. Como, Italy (2009)
7. Tiller M.: Introduction to Physical Modeling with Modelica, 2nd edn. Kluwer Academic Publishers, Massachusetts (2004)
8. Astrom, K., Hagglund, T.: PID Controllers: Theory, Design, and Tuning, 2nd edn. Instrument Society of America, North Carolina (1995)
9. Brater, E., King, H., Lindell, J.: Handbook of Hydraulics, 7th edn. Mc Graw Hill, New York (1996)
10. Dynasim AB. Dymola dynamic modeling laboratory user manual, Sweden (2010)
11. Jensen, J.: Dynamic modeling of thermo-fluid systems with focus on evaporators for refrigeration. Ph.D. thesis, Department of Mechanical Engineering, Technical University of Denmark, Kongens Lyngby (2003)
12. Abdelati, M., Felgner, F., Frey, G.: Modeling wastewater pumping stations for cost-efficient control. In: 17th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA 2012), Kraków, Poland (2012)
13. Felgner, F., Exel, L., Frey, G.: Component-oriented ORC plant modeling for efficient system design and profitability prediction. In: Proceedings of the IEEE/IES International Conference on Clean Electrical Power (ICCEP 2011), pp. 196–203. Ischia, Italy (2011)
14. Sonntag, C., Kölling, M., Engell, S.: Sensitivity-based predictive control of a large-scale supermarket refrigeration system. In: International Symposium on Advanced Control of Chemical Processes (ADCHEM), pp. 354–359. Istanbul, Turkey (2009)

Chapter 19

A System Identification Framework for Modeling Complex Combustion Dynamics Using Support Vector Machines

Vijay Manikandan Janakiraman, XuanLong Nguyen, Jeff Sterniak
and Dennis Assanis

Abstract Machine Learning is being widely applied to problems that are difficult to model using fundamental building blocks. However, the application of machine learning in powertrain modeling is not common because existing powertrain systems have been simple enough to model using simple physics. Also, black box models are yet to demonstrate sufficient robustness and stability features for widespread powertrain applications. However, with emergence of advanced technologies and complex systems in the automotive industry, obtaining a good physical model in a short time becomes a challenge and it becomes important to study alternatives. In this chapter, support vector machines (SVM) are used to obtain identification models for a gasoline homogeneous charge compression ignition (HCCI) engine. A machine learning framework is discussed that addresses several challenges for identification of the considered system that is nonlinear and whose region of stable operation is very narrow.

Keywords Support vector machine · Identification · Combustion · Homogeneous charge compression ignition · HCCI · Neural networks · Nonlinear regression · Engine model · Control model

V. M. Janakiraman (✉) · X. Nguyen
University of Michigan, Ann Arbor, MI, USA
e-mail: vijai@umich.edu

X. Nguyen
e-mail: xuanlong@umich.edu

J. Sterniak
Robert Bosch LLC, Farmington, MI, USA
e-mail: Jeff.Sterniak@us.bosch.com

D. Assanis
Stony Brook University, NY, USA
e-mail: dennis.assanis@stonybrook.edu

19.1 Introduction

In recent years, the requirements on automotive performance, emissions and safety have become increasingly stringent. In spite of advanced concepts entering the automotive industry, achieving fuel economy, emission and cost targets simultaneously still remains an arduous task. Homogeneous Charge Compression Ignition (HCCI) engines gained the spotlight from traditional spark ignited and compression ignited engines owing to the ability to reduce emissions and fuel consumption significantly [1]. In spite of its known advantages, HCCI combustion poses several challenges for use in an automobile. A major challenge is achieving stable combustion over a wide operating range. HCCI control is a challenging problem and a predictive model is typically used to make decisions [2]. Hence it becomes important to develop accurate HCCI models that can operate with manageable computational demand so that it can be implemented on-board for controls and diagnostics purposes. HCCI combustion is characterized by complex nonlinear chemical and thermal dynamics, which necessitates significant development times and expert labor to develop models using physics. Also, the model may be required to predict several steps ahead of time with reasonable accuracy for analysis and optimal control. Hence a key requirement is to develop a model quickly that can capture the required dynamics for control purposes and has the potential to be implemented on-board.

Identification could be an effective alternative for developing control oriented models quickly. Identification models have the advantages that simple models can be built quickly without much expert knowledge about the system. Also, the models developed based on real measurements can capture the behavior of the system up to the desired level of fidelity. However, such black box models rarely give insights into the system. Also, if the system to be identified is nonlinear having many degrees of freedom with a very narrow stable region of operation, it becomes an almost impossible task to identify the system completely. Hence it is not surprising that related literature is scarce for HCCI identification. A subspace based identification was the only reported work [3] where linear models were developed for HCCI model predictive control. However, for stable engines like spark ignited and compression ignition engines, a few applications of neural networks for engine modeling and control have been reported [4]. This chapter [5] aims to be the first application of a machine learning approach for nonlinear identification of the HCCI combustion.

For the HCCI identification problem considered in this chapter, a Support Vector Machine based regression was selected for its good approximation capabilities and robustness to fit nonlinear data [6, 7]. The application of SVR to system identification [8–10], time series modeling [11] and predicting chaotic behavior [12] has been reported in the literature though major practical implementations were less abundant [8–10]. Also, when SVR is trained on real-world data, it represents the real system and makes no simplifying assumptions of the underlying process. The dynamics of sensors, actuators and other complex processes, which are usually overlooked/hard to model using physics, can be captured using the identification method. In addition, for a system like the combustion engine, prototype hardware is typically available,

and sufficient experimental data can be collected. However, design of experiments and data preprocessing becomes a challenge for nonlinear and partially stable system like the HCCI engine.

This chapter is organized as follows. The basic idea and formulation of the SVR model is presented in Sect. 19.2, HCCI system description and data collection in Sect. 19.3. A framework for HCCI identification using SVR has been discussed in Sect. 19.4 followed by performance evaluations and model validation results.

19.2 Support Vector Regression for Nonlinear Identification

The Support Vector Regression (SVR) [13] was developed as an extension to the Support Vector Machines (SVM) originally developed for classification. The SVR model approximates the given input-output data by forming an error boundary (error tube) [13] around the data by solving a convex constrained optimization problem. An important property of the SVR method is that the obtained model could be a sparse representation of the nonlinear system which can have benefits in terms of storage.

A generic nonlinear identification using the nonlinear auto regressive model with exogenous input (NARX) is considered as follows

$$y(k) = f[u(k - 1), \dots, u(k - n_u), y(k - 1), \dots, y(k - n_y)] \tag{19.1}$$

where $u(k) \in \mathbb{R}^{u_d}$ and $y(k) \in \mathbb{R}^{y_d}$ represent the inputs and outputs of the system respectively, k represents the discrete time index, $f(\cdot)$ represents the nonlinear function mapping specified by the model, n_u, n_y represent the number of past input and output samples required (order of the system) for prediction while u_d and y_d represent the dimension of inputs and outputs respectively. Let x represent the augmented input vector obtained by appending the input and output measurements from the system.

$$x = [u(k - 1), \dots, u(k - n_u), y(k - 1), \dots, y(k - n_y)]^T \tag{19.2}$$

The input measurement sequence can be converted to the form of training data as required by SVR

$$\{(x_1, y_1), \dots, (x_n, y_n)\} \in (\mathcal{X}, \mathcal{Y}) \tag{19.3}$$

where \mathcal{X} denotes the space of the input features (Here $\mathcal{X} = \mathbb{R}^{u_d n_u + y_d n_y}$ and $\mathcal{Y} = \mathbb{R}$). The goal of SVR is to approximate the underlying input-output function mapping $f(\cdot)$ by minimizing a risk functional with respect to the model parameters

$$R(w) = \frac{1}{n} \sum_{i=1}^n L(y_i - \hat{y}_i(x, w)) + \frac{1}{2} w^T w \tag{19.4}$$

where $\hat{y}(x, w)$ represents the model prediction given by

$$\hat{y}(x, w) = \langle w, \phi(x) \rangle + b \tag{19.5}$$

Here, $w \in \mathbb{R}^{u_d n_u + y_d n_y}$ and $b \in \mathbb{R}$ represents the model parameters, ϕ is a function that transforms the input variables to a higher dimension feature space \mathcal{H} and $\langle \cdot, \cdot \rangle$ represents inner product in \mathcal{H} . The first term of Eq. (19.4) represents the error minimizing term while the second term accounts for regularization. The SVR model deals only with the inner products of ϕ and a kernel function can be defined that takes into account the inner products implicitly as

$$K(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle \tag{19.6}$$

The function ϕ is not required to be known but any kernel function that satisfies the Mercer’s condition such as radial basis functions (19.7), polynomial (19.8) and sigmoidal functions (19.9) can be used. In this study, along with the above kernels, the linear kernel (pure inner product) (19.10) is also used. The kernel transforms the input variables to a high dimension space \mathcal{H} and aids in converting a nonlinear map in the $\mathcal{X} - \mathcal{Y}$ space to a linear map in \mathcal{H} space. This is known as *kernel trick* in the literature.

$$K(x_i, x_j) = e^{-\omega \|x_i - x_j\|^2}, \omega > 0 \tag{19.7}$$

ccc

$$K(x_i, x_j) = (\omega \langle x_i, x_j \rangle + c_0)^{deg}, \omega > 0 \tag{19.8}$$

$$K(x_i, x_j) = \tanh(\omega \langle x_i, x_j \rangle + c_0), \omega > 0 \tag{19.9}$$

$$K(x_i, x_j) = \langle x_i, x_j \rangle = x_i^T x_j \tag{19.10}$$

The ν -SVR model is considered in this study, as the tradeoff between model complexity and accuracy (controlled by ν) can be tuned to the required accuracy and sparseness. Sparseness is the ratio of the number of support vectors to the total number of data observations in the model. The following ϵ -insensitive loss function is used.

$$L(y - \hat{y})_\epsilon = \begin{cases} 0 & \text{if } |y - \hat{y}| \leq \epsilon \\ \epsilon - |y - \hat{y}| & \text{otherwise} \end{cases} \tag{19.11}$$

The goal of SVR training is to determine the optimal model parameters (w^*, b^*) that minimizes the risk function (19.4) which reduces to minimizing the slack variables ζ and ζ^* which leads into the following optimization problem.

$$\min_{w, b, \epsilon, \zeta_i, \zeta_i^*} \frac{1}{2} w^T w + C(\nu \epsilon + \frac{1}{n} \sum_{i=1}^n (\zeta_i + \zeta_i^*)) \tag{19.12}$$

$$\text{subjected to } \begin{cases} y_i - (\langle w, \phi(x_i) \rangle + b) \leq \epsilon + \zeta_i \\ (\langle w, \phi(x_i) \rangle + b) - y_i \leq \epsilon + \zeta_i^* \\ \zeta_i, \zeta_i^*, \epsilon \geq 0 \end{cases} \quad (19.13)$$

for $i = 1, \dots, n$. It should be noted that the slack variables take values of zero when the points lie inside the error tube. Also, separate slack variables ζ and ζ^* are assigned for points lying outside the error tube on either side of the function. The above optimization problem is usually referred as the primal problem and the variables w, b, ζ, ζ^* and ϵ are the primal variables. In the above formulation (19.12), ϵ is considered as a variable to be optimized along with the model parameters. This allows ν to set a lower bound on the fraction of data points used in parameterizing the model [14] and hence by tuning ν one can achieve a tradeoff between model complexity (sparseness) and accuracy. A value of ν close to unity tries to shrink the ϵ tube and reduce sparseness (all data points become support vectors) while reducing ν close to zero will result in a sparse model (very few data points are used in model parametrization) with possible under-fitting. This flexibility is the prime reason for selecting the ν -SVR algorithm for this study.

The lagrangian can be formulated as follows

$$\begin{aligned} L(w, b, \zeta, \zeta^*, \epsilon, \alpha, \alpha^*, \beta, \beta^*, \gamma) = & \frac{1}{2} w^T w + C(\nu\epsilon + \frac{1}{n} \sum_{i=1}^n (\zeta_i + \zeta_i^*)) \\ & + \sum_{i=1}^n \alpha_i (y_i - (\langle w, \phi(x_i) \rangle + b) - \epsilon - \zeta_i) \\ & + \sum_{i=1}^n \alpha_i^* ((\langle w, \phi(x_i) \rangle + b) - y_i - \epsilon - \zeta_i^*) \\ & - \sum_{i=1}^n (\beta_i \zeta_i + \beta_i^* \zeta_i^*) - \sum_{i=1}^n \gamma \epsilon \end{aligned} \quad (19.14)$$

where $\alpha, \alpha^*, \beta, \beta^*, \gamma$ are the lagrange multipliers or the dual variables. Applying first order optimality conditions we can convert the primal problem (Eqs. (19.12) and (19.13)) to the following dual optimization problem

$$\max_{\alpha_i, \alpha_i^*} \sum_{i=1}^n (\alpha_i^* - \alpha_i) y_i - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n (\alpha_i^* - \alpha_i) (\alpha_j^* - \alpha_j) K(x_i, x_j) \quad (19.15)$$

$$\text{subjected to } \begin{cases} \sum_{i=1}^n (\alpha_i^* - \alpha_i) = 0 \\ \sum_{i=1}^n (\alpha_i^* + \alpha_i) \leq \nu C \\ 0 \leq \alpha_i \leq \frac{C}{n} \\ 0 \leq \alpha_i^* \leq \frac{C}{n} \end{cases} \quad (19.16)$$

Table 19.1 Specifications of the experimental HCCI engine

Engine type	4-stroke In-line
Fuel	Gasoline
Displacement	2.0 L
Bore/Stroke	86/86
Compression ratio	11.25:1
Injection type	Direct injection
Valvetrain	Variable valve timing with hydraulic cam phaser (0.25 mm constant lift, 119° constant duration and 50° crank angle phasing authority)
HCCI strategy	Exhaust recompression using negative valve overlap

for $i = 1, \dots, n$. Solving the dual problem yields α_i and α_i^* giving the following SVR model

$$f(x) = \sum_{i=1}^n (\alpha_i^* - \alpha_i) K(x_i, x) + b \quad (19.17)$$

where ϵ and b can be determined using (19.13). This is the well known SVR model and the following are some known properties. The parameter w can be completely described as a linear combination of functions of the training data (x_i). The model is independent of the dimensionality of \mathcal{X} and the sample size n and the model can be described by dot products between the data.

19.3 HCCI System and Experiments

The data for system identification is collected from a variable valve timing gasoline HCCI engine whose specifications are listed in Table 19.1. HCCI is achieved by auto-ignition (without spark initiation) of the gas mixture in the cylinder. High volumes of exhaust gas residuals (EGR) are inducted into the cylinder to maintain low combustion temperatures for reduced nitrous oxide emissions. A variable valve timing capability allows the closure of the exhaust valve and opening of the intake valve to be modified so as to create a negative valve overlap (NVO) to trap the desired quantity of EGR in the cylinder. The fuel injection also happens during the NVO event. The EGR and fuel injected directly influences the temperature and concentration of the gas mixture entering the next combustion cycle. The pressure trace during one combustion cycle along with valve events and fuel injection events are shown in Fig. 19.1. The smaller pressure peak represents the NVO event. The variables that indicate HCCI performance are net mean effective pressure (IMEP), combustion

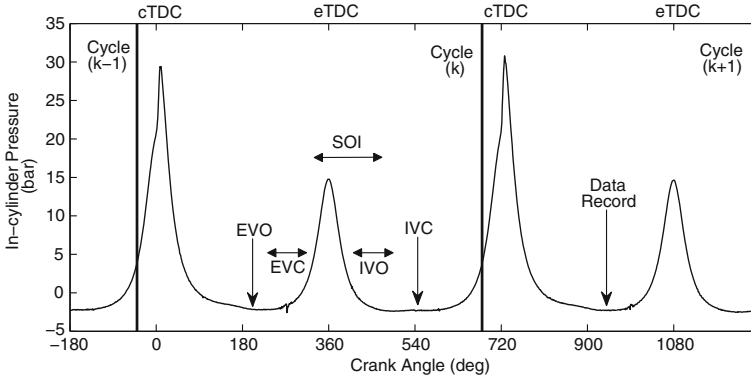


Fig. 19.1 HCCI engine pressure trace showing cycle definition, actuator ranges of intake valve opening (IVO), exhaust valve closing (EVC), start of injection (SOI). Fixed events of exhaust valve opening (EVO) and intake valve closing (IVC) and crank angle at data recording are also shown

phasing measured as crank angle at 50% mass fraction burned (CA50), combustion roughness measured by maximum rate of pressure rise (R_{max}) in the cylinder and equivalent air-fuel ratio (λ) defined as

$$\lambda = \frac{(A/F)}{(A/F)_s} \quad (19.18)$$

where A/F = mass of air per cycle/mass of fuel per cycle and $(A/F)_s = (A/F)$ at stoichiometric condition.

The design of excitation signals and data measurement are well explained in [5]. For the HCCI system, the fuel mass (FM), crank angle at intake valve opening (IVO), crank angle at exhaust valve closing (EVC) and crank angle at start of injection (SOI) were considered as input signals following the amplitude modulated pseudo random binary sequence (A-PRBS) pattern. Engine variables such as IMEP, CA50, R_{max} , λ , intake manifold temperature (T_{in}), intake manifold pressure (P_{in}), exhaust manifold pressure (P_{ex}), mass flow rate at intake (\dot{m}_{in}), exhaust manifold temperature (T_{ex}), coolant temperature (T_c), λ etc. are measured using the on-board engine control unit on a combustion cycle basis.

19.4 SVR Modeling Framework

This section details the training procedure using the SVR method described in Sect. 19.2. A good definition of the regressor variables as well as an optimal selection of hyper-parameters are necessary for developing robust models. Also, data preprocessing is required so that the data can be efficiently used for machine learning. This section discusses the above tasks specifically for the HCCI identification problem.

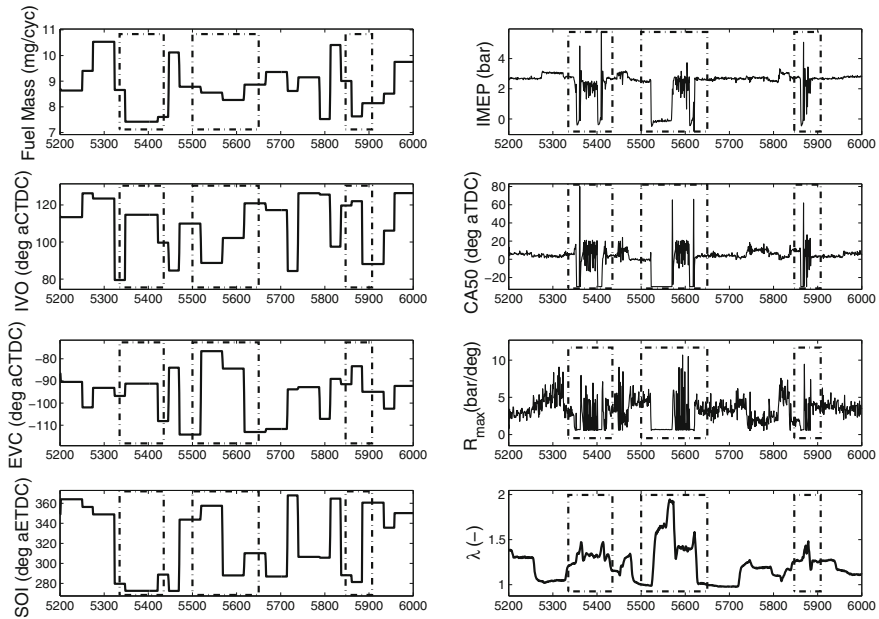


Fig. 19.2 A-PRBS inputs and outputs showing misfire regions

19.4.1 Data Preprocessing

For any machine learning task, the data needs to be appropriately scaled and centered. This includes normalization of all data to lie between -1 and $+1$ to ensure the model parameters to be of the same order improving numerical stability. For the HCCI system, additional preprocessing needs to be done before the data becomes meaningful for learning. The unstable nature of HCCI and designing experiments to reduce unstable data collection has been discussed in [5]. However, it is not possible to eliminate misfires during the experiments. Hence, the data corresponding to misfires needs to be removed before using it for learning. Figure 19.2 shows misfire regions along with stable HCCI measurements. From a physical point of view, during a misfire, the cylinder is filled with unburnt (or partially burnt) fuel along with some exhaust from the previous cycle. Hence the value of λ increases indicating high concentration of air and less fuel. Also, the IMEP (indicator for work output) drops to zero as combustion failed to occur. The value of CA50 drops to -30 (set by the measurement system during no combustion) while R_{max} drops to 0 as there is no pressure rise. During a regular combustion event, some of the exhaust gas is trapped by the NVO for the next cycle but when a misfire occurs, there is not enough exhaust gas in the NVO trapped mixture for the subsequent cycles. Hence even if the next input in the PRBS sequence is stable during regular operation, after a misfire the

Table 19.2 The minimum and maximum values of regression variables x and y determined based on expert knowledge for HCCI conditions

	x										y			
	FM	IVO	EVC	SOI	T_c	T_{in}	T_{ex}	P_{in}	P_{ex}	(\dot{m}_{in})	IMEP	CA50	R_{max}	λ
Min	6.9	78	-119	270	89.00	49.00	349.00	0.85	1.00	91.00	0.50	1.00	0.00	1.00
Max	11.2	128	-69	380	93.00	70.00	440.00	1.40	1.20	370.00	4.00	11.00	6.00	2.00

combustion behavior does not represent regular HCCI operation. Hence along with the misfire data, some of the post-misfire data needs to be eliminated.

It is not trivial that a portion of data can be removed from a time series and still maintain the time connection. However the combustion variables can be used to determine the validity of regular HCCI operation. The misfire itself can be labeled using a combination of IMEP and CA50. The identification model structure (19.1) and regressor definition (19.2) allows time series data to be converted to feature vectors to be mapped on to the output variable. Hence after the regressor conversion the data can be treated as static (no time connection between observations). It is for the same reason that it becomes possible to use a static regression model like the SVR for time series data. The static data $\{(x_1, y_1), \dots, (x_n, y_n)\}$ can be filtered using expert knowledge about the engine. The minimum and maximum limits of the vectors x and y are used to identify and eliminate data that might not represent HCCI combustion. The minimum and maximum limits of the variables are shown in Table 19.2. The data is normalized based on the variable limits to lie between -1 and $+1$. Any pair (x, y) whose normalized value lies outside -1 and $+1$ is considered inappropriate data and removed from the data set.

19.4.2 Regressor Variables

In an identification problem, the regressor variables can include several input variables (features) but not all of them may be required. For instance, in the HCCI combustion engine, several sensors measure temperatures, pressures and flows in and around the combustion chamber. However, it is understood from physics that only a few are responsible for the variables of interest. Hence the regressor variables can be defined using expert knowledge about the system. The primary variables affecting HCCI combustion include temperature, concentration of gas mixture at the time of intake valve closing (see Fig. 19.1). Since these quantities are not measured directly, the secondary variables that affect the primary variables are included in the regressor definition. The temperature, pressure and air flow rate before the intake manifold, the temperature at the exhaust manifold along with the injected fuel mass, valve events, injection events and coolant temperature has a major influence on the temperature and concentration of the gas mixture in the cylinder. Hence the regressor variable is defined to include the history of the measurements of the above quantities.

19.4.3 Model Selection

For each output, the model has four hyper-parameters namely the system order (number of past data from history, $n_o = n_u = n_y$, assumed to be the same), the cost parameter C , kernel parameter ω and SVR parameter ν . To obtain a model that generalizes well and captures the right order of dynamics, the above hyper-parameters need to be optimized based on cross-validation. The data set comprising of (x, y) is divided into a training set that constitutes 70% of the data while the remaining 30% is separated out for testing. Furthermore, the training data is divided into validation training and validation testing data sets for tuning the model hyper-parameters. The validation data consists of a randomly sampled subset of the training data (20% of the training set). The testing data set is never seen by the model during the training and validation phases.

The parameter ν determines the tradeoff between the sparseness and accuracy of the models. An optimal value of ν results in the minimum model parameters required for the given accuracy level. The cost parameter C determines the relative importance given to the outliers and hence the sensitivity to measurement noise. A large value of C tries to fit the model for outliers thereby over-fitting the data. The kernel parameter ω is required to be tuned for the same reason of having good generalization. The system order n_o determines the number of previous measurements required to predict the future output. An optimal value of the order represents the system dynamics correctly and a large value not only makes the model complex by increasing the dimension of x but also gives a bad prediction of the system's response.

A full grid search was performed over all possible combinations of n_o, C, ω, ν in the selected range and the combination that had the minimum validation error was chosen as the optimal hyper-parameters. A detailed procedure on tuning the model hyper-parameters using cross-validation is explained in [5]. Table 19.3 lists the best combination of hyper-parameters for IMEP, CA50, R_{max} and λ which had the minimum validation errors. The training times were very long for polynomial kernel models while the training time for linear, sigmoidal and gaussian kernels were comparable. Hence the models with linear, sigmoidal and gaussian kernels were retrained with the optimal hyper-parameters on the entire training data set while the ones with polynomial kernels were retrained on a subset of the training set.

19.4.4 SVR Prediction Results

The SVR models are simulated with the unseen test data set and performance of the models are measured using root mean squared error (RMSE) given by (19.19). Note that the RMSE is different from the loss function (19.11) used in SVR modeling. Table 19.4 compares the performance of the models in terms of training and testing RMSE, memory units and the number of parameters required. The memory units represent the number of sequential data history that needs to be stored in the memory as defined by x and y . The number of parameters represent the sparsity as controlled

Table 19.3 The optimal values of system order ($n_o = n_u = n_y$, assumed to be the same), cost parameter C , kernel parameter ω and SVR parameter ν determined using cross-validation from the range of listed values

Kernel type	Hyper-parameter	IMEP	CA50	R_{max}	λ	Range
Linear	n_o	2	5	5	4	{1, 2, 3, 4, 5}
	C	1	0.1	0.1	100	{0.01, 0.1, 1, 10, 100}
	ν	0.2	0.6	0.4	0.2	{0.1, 0.2, ..., 1}
Polynomial of degree 2	n_o	2	3	4	4	{1, 2, 3, 4, 5}
	ω	1	1	1	1	{0.001, 0.01, 0.1, 1, 10, 100}
	C	1	1	10	1	{0.01, 0.1, 1, 10, 100}
Polynomial of degree 3	ν	0.4	0.4	0.3	0.5	{0.1, 0.2, ..., 1}
	n_o	2	2	1	3	{1, 2, 3, 4, 5}
	ω	0.5	0.2	1	0.5	{0.01, 0.1, 0.2, ..., 1, 5, 10}
Sigmoidal	C	0.5	0.5	1	1.5	{0.01, 0.1, 0.2, ..., 2}
	ν	0.5	0.4	0.4	0.3	{0.1, 0.2, ..., 1}
	n_o	2	5	5	3	{1, 2, 3, 4, 5}
Gaussian	ω	0.01	0.01	0.01	0.01	{0.001, 0.01, 0.1, 1, 10, 100}
	C	1	1	1	1	{0.01, 0.1, 1, 10, 100}
	ν	0.6	0.4	0.4	0.4	{0.1, 0.2, ..., 1}
Gaussian	n_o	2	3	1	2	{1, 2, 3, 4, 5}
	ω	0.1	0.1	1	0.1	{0.001, 0.01, 0.1, 1, 10, 100}
	C	1	1	1	1	{0.01, 0.1, 1, 10, 100}
	ν	0.3	0.4	0.4	0.3	{0.1, 0.2, ..., 1}

by the parameter ν .

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2} \tag{19.19}$$

It can be seen from Table 19.4 that all the models capture the dynamics of IMEP, CA50, R_{max} and λ to a reasonable accuracy but with different memory and storage requirements. Gaussian kernel outperforms all the other kernels in terms of both achieving minimum accuracy as well as with relatively low memory and storage requirements. This can also be seen in Table 19.3 where gaussian kernel identifies the system with a relatively lower order (n_o) for all the response variables considered.

The performance of all the kernels are comparable for IMEP with similar memory requirements. Indeed, the linear kernel uses less parameters compared to the gaussian kernel. This may be attributed to the simpler mechanism behind IMEP which strongly depends on the fuel mass injected [2] and can be potentially identified using linear models. It should be noted that the polynomial kernel models are tested on a smaller data set and hence the number of parameters are low. Overall, the gaussian kernel is chosen as the appropriate one for modeling the HCCI engine behavior. The noise distribution for the engine output variables are shown in Fig. 19.3. It can be seen that the noise amplitude levels are an indication of the expected performance error.

Table 19.4 Performance comparison of SVR

Kernel type		IMEP	CA50	R_{max}	λ
Linear	Training RMSE	0.0686	0.2621	0.2571	0.0173
	Testing RMSE	0.0735	0.2678	0.2555	0.0173
	n_m	20	50	50	50
	n_p	73502	330825	280335	168212
Polynomial of degree 2	Training RMSE	0.0693	0.2451	0.2398	0.0127
	Testing RMSE	0.0663	0.2793	0.3098	0.0155
	n_m	20	30	40	40
	n_p	46486	70455	79200	125488
Polynomial of degree 3	Training RMSE	0.0707	0.2651	0.2494	0.0141
	Testing RMSE	0.0671	0.2691	0.2583	0.0173
	n_m	20	20	10	30
	n_p	57068	45012	22517	56265
Sigmoidal	Training RMSE	0.0742	0.3082	0.3103	0.0245
	Testing RMSE	0.0693	0.3093	0.3055	0.0245
	n_m	20	50	50	30
	n_p	218372	220330	279400	192456
Gaussian	Training RMSE	0.0632	0.2396	0.2079	0.0173
	Testing RMSE	0.0608	0.2525	0.2181	0.0173
	n_m	20	30	10	20
	n_p	113652	149787	69861	102982

n_m and n_p represents number of memory units and number of parameters (support vectors) required by the models

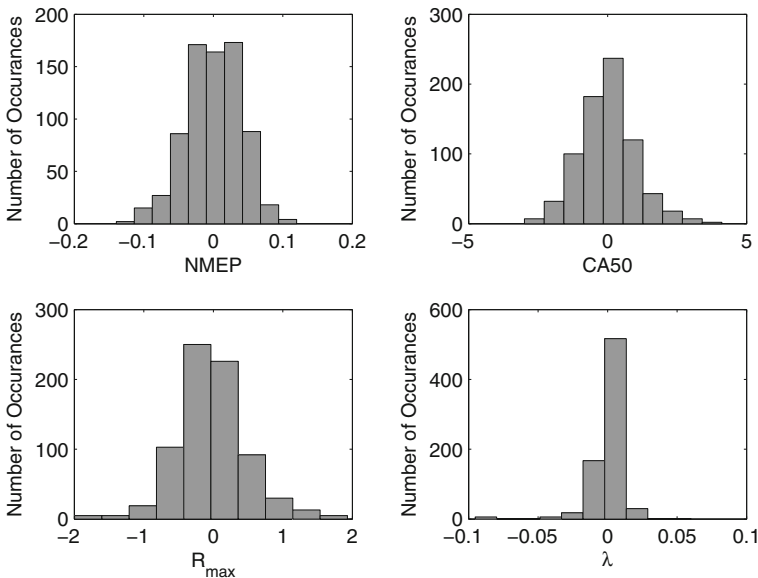


Fig. 19.3 Noise distribution of the engine output variables

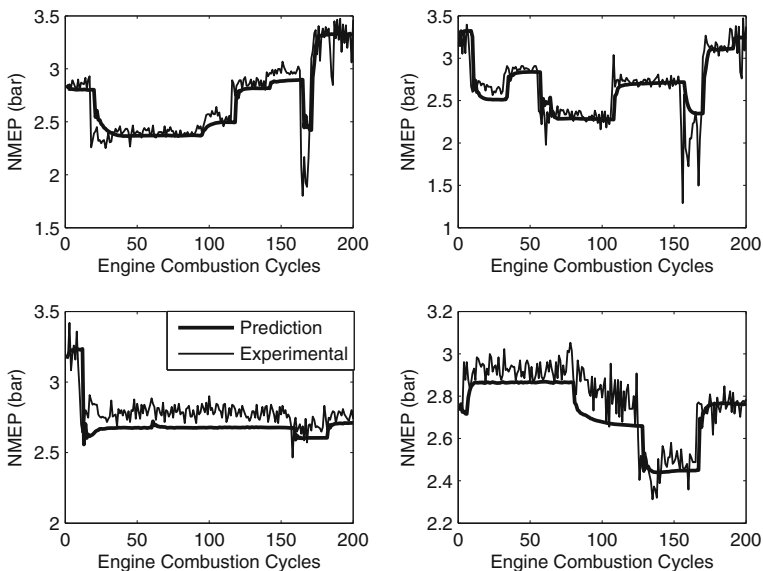


Fig. 19.4 Comparison of IMEP (engine output and SVR prediction)

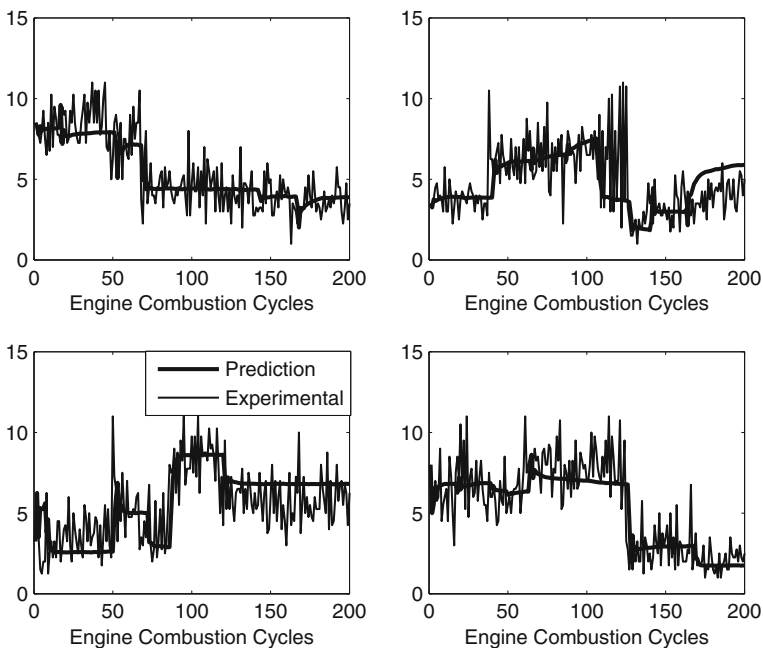


Fig. 19.5 Comparison of CA50 (engine output and SVR prediction)

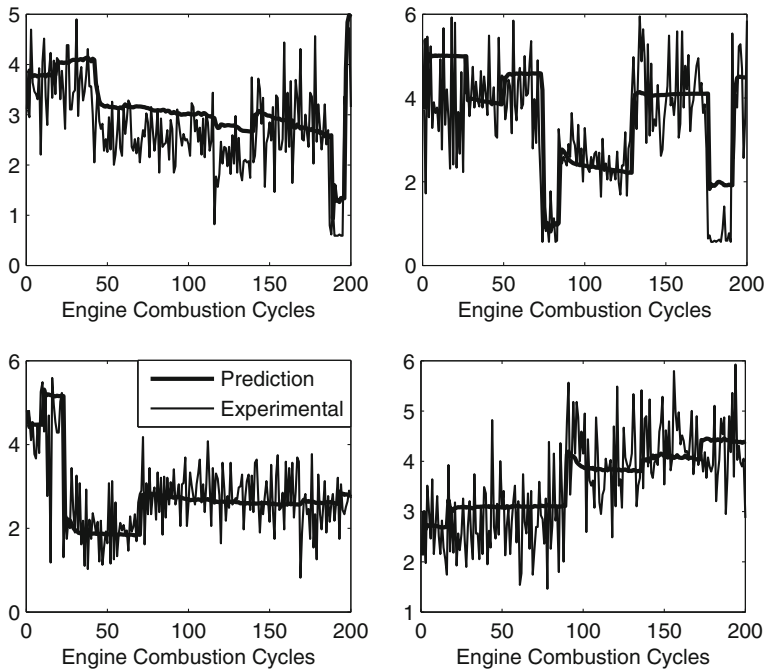


Fig. 19.6 Comparison of Maximum rate of pressure rise (engine output and SVR prediction)

For instance, λ is a filtered signal within the engine control unit and hence noise amplitude is lower than the other signals, hence the error magnitude for λ as shown in Table 19.4. CA50 on the other hand, has a higher noise magnitudes and hence high estimation error. The noise distribution is gaussian and hence the ϵ -insensitive loss function (19.11) also appears to be appropriate for the data.

19.4.5 Multi-Step-Ahead Predictions

In order to observe the multi-step-ahead predictions for the selected gaussian kernel model, a completely separate data set is used wherein the input trajectories are given to the model along with the initial conditions of the outputs (delay initial conditions). The response of the model is fed back to construct the regressor for subsequent cycles representing a discrete dynamic system simulation. The 200 cycle ahead predictions of IMEP, CA50, R_{max} and λ are compared against measured data from the engine in Figs. 19.4, 19.5, 19.6 and 19.7 respectively. Each figure shows predictions of an output variable for four different cases as validations in different operating conditions. It can be observed from the figures that the model can simulate the HCCI dynamics to a good accuracy and can be used as predictors of transients multiple steps ahead

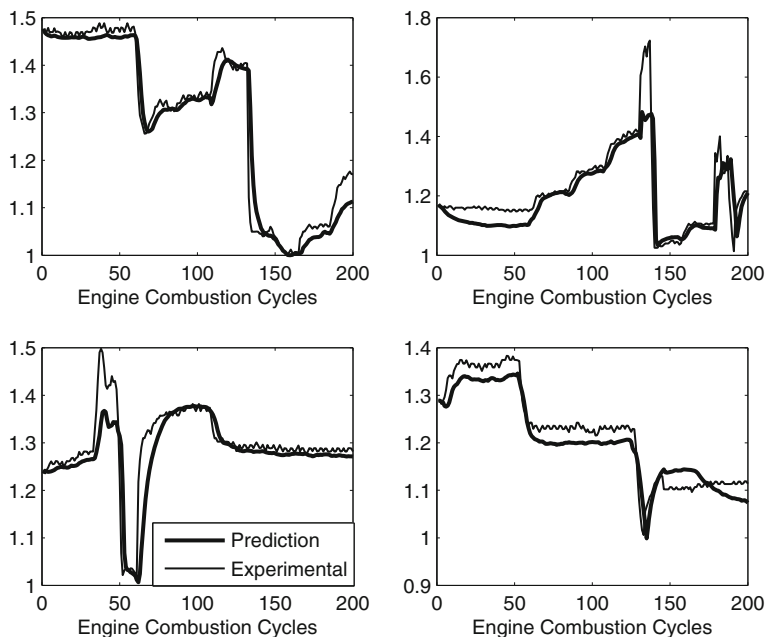


Fig. 19.7 Comparison of Lambda (engine output and SVR prediction)

in time. It can also be observed that both the steady state values and the transients are well captured by the models except for a few regions where there is a bias offset owing to poor approximations. Lack of excitations near such input combinations could be a reason for the bad predictions of the model. The ν parameter in Table 19.3 shows that only a small fraction of the training data set is used to represent the model efficiently and these data observations constitute the support vectors in this method.

19.5 Conclusions and Future Work

Support Vector Machines are one of the state of the art methods for robust learning but the application of SVR to nonlinear system identification is not abundant in spite of its attractive properties. In this chapter¹ a complex nonlinear dynamic system such

¹ Acknowledgements and disclaimer: This material is based upon work supported by the Department of Energy [National Energy Technology Laboratory] under Award Number(s) DE-EE0003533. This work is performed as a part of the ACCESS project consortium (Robert Bosch LLC, AVL Inc., Emitec Inc.) under the direction of PI Hakan Yilmaz, Robert Bosch, LLC. This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed,

as the HCCI combustion engine is identified using ν -SVR method. Data collection within the stable boundary of HCCI combustion has been accomplished by pre-screening the input sequence using the steady state model of the engine and by using a feedback loop to minimize unstable excitations. A machine learning framework has been described giving insights into experiment design for the partially stable system, data pre-screening to handle the corrupted data, a cross-validation based systematic model selection, regressor definition etc.

A comparative study of different kernels—linear, quadratic, cubic, sigmoid and gaussian kernels has been performed showing benefits of gaussian kernel for the HCCI system in terms of accuracy and memory/storage requirements. The dynamics of IMEP, CA50, R_{max} and λ of naturally aspirated gasoline HCCI combustion at constant speed is modeled using SVR to a good accuracy. Distinguishing features of ν -SVR including global optimality and sparseness make the method very attractive compared to traditional neural networks based identification. Future research would focus on controller development using the SVR models to analyze the suitability and effectiveness compared to both existing physics based and neural network based controllers.

References

1. Aoyama, T., Hattori, Y., Mizuta, J., Sato, Y.: An experimental study on premixed-charge compression ignition gasoline engine. In: SAE Technical Paper 960081 (1996).
2. Ravi, N., Roelle, M.J., Liao, H.H., Jungkunz, A.F., Chang, C.F., Park, S., Gerdes, J.C.: Model-based control of hcci engines using exhaust recompression. *IEEE Trans. Control Syst. Technol.* **13**(1), 5 (2009)
3. Bengtsson, J., Strandh P., Johansson R., Tunestal P., Johansson, B.: Model predictive control of homogeneous charge compression ignition (HCCI) engine dynamics. In: 2006 IEEE International Conference on Control Applications (2006)
4. Bloch, G., Lauer, F., Colin, G.: On learning machines for engine control. *Comput. Int. Autom. Appl.* **13**(2), 125–144 (2008)
5. Janakiraman, V.M., Sterniak, J., Assanis, D.: Support vector machines for identification of HCCI combustion dynamics. *International Conference on Informatics in Control, Automation and Robotics (ICINCO)*, Rome, In (2012)
6. Hammer, B., Gersmann, K.: A note on the universal approximation capability of support vector machines. In: *Neural Processing Letters*. Kluwer Academic Publishers, Boston (2003)
7. Clarke, S.M., Griebisch, J.H., Simpson, T.W.: Analysis of support vector regression for approximation of complex engineering analyses. *J. Mech. Des.* **127**, 1077–1087 (2005)
8. Wang, X., Chen, DuZ, J., Pan, F.: Dynamic modeling of biotechnical process based on online support vector machine. *J. Comput.* **4**(3), 251–258 (2009). <http://ojs.academypublisher.com/index.php/jcp/article/view/0403251258>

(Footnote 1 continued)

or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

9. Chitralkha, S.B., Shah, S.L.: Application of support vector regression for developing soft sensors for nonlinear processes. *The Can. J. Chem. Eng.* **88**(6), 899–911 (2010)
10. Chitralkha, S.B., Shah, S.L.: *Application of Support Vector Regression for Developing Soft Sensors for Nonlinear Processes*. Wiley, New York (2010)
11. Müller, K.R., Smola, A.J., Ratsch, G., Scholkopf, B., Kohlmorgen, J., Vapnik, V.: Predicting time series with support vector machines. In: *Artificial Neural Networks—ICANN'97*. Springer, Berlin (1997)
12. Sun, J., Zhou, Y., Bai, Y., Luo, J.: Nonlinear noise reduction of chaotic time series based on multi-dimensional recurrent least squares support vector machines. In: *Neural Information Processing, LNCS*. Springer, Heidelberg (2006)
13. Drucker, H., Burges, C.J.C., Kaufman, L., Smola, A., Vapnik, V.: *Support Vector Regression Machines* (1996)
14. Schölkopf, B., Smola, A.J., Williamson, R.C., Bartlett, P.L.: New support vector algorithms. In: *Neural Computation*. MIT Press, Cambridge (2000)

Author Index

A

Abdelati, M., 281
Assanis, D., 298

B

Beeck, K. V., 252
Bokor, J., 115
Bondarenko, D., 198
Briot, S., 198
Brüggemann, B., 147
Brunner, M., 147

C

Chiara, B., 265
Cho, S.-B., 103
Chtourou, M., 183
Cuzzola, S., 265

D

Deflorio, F., 265

F

Felgner, F., 281
Fernández, L., 132
Frey, G., 282
Furet, B., 198

G

Gáspár, P., 115
Gil, A., 132
Goedemé T., 252

H

Hammouda, L., 183

J

Janakiraman, V., 298

K

Kaaniche, K., 183
Kim, J., 35
Klimchik, A., 198
Koumboulis, F., 67

L

Lee, Y.-S., 103

M

Maier, A., 19
Marino, R., 3
Mekki, H., 183

N

Nedevschi, S., 235
Nguyen, X., 298
Niggemann, O., 19
Ntellis, A. S., 67

O

Onat, A., 167
Ozkan, M., 167

P

Palkovics, L., 115

Parigi, G., 83
Pashkevich, A., 198
Pau, D., 84
Piastra, M., 84

Q

Qayyum, U., 35

R

Rödönyi, G., 115
Reinoso, O., 132

S

Santos, C., 218
Scalzi, S., 3
Schulz, D., 147

Semenkin, E., 51
Semenkina, M., 51
Silva, A., 218
Silva, J., 218
Skarpetis, M., 67
Sterniak, J., 298
Stramieri, A., 83

T

Tack, T., 19
Tomei, P., 3
Tuytelaars T., 252

V

Valiente, D., 132
Vatavu, A., 235
Verrelli, C., 3