
Springer Tracts in Advanced Robotics

Volume 73

Editors: Bruno Siciliano · Oussama Khatib · Frans Groen

Peter Corke

Robotics, Vision and Control

Fundamental Algorithms in MATLAB®

With 393 Images

Additional material is provided at www.petercorke.com/RVC

Professor Bruno Siciliano, Dipartimento di Informatica e Sistemistica, Università di Napoli Federico II,
Via Claudio 21, 80125 Napoli, Italy, E-mail: siciliano@unina.it

Professor Oussama Khatib, Artificial Intelligence Laboratory, Department of Computer Science,
Stanford University, Stanford, CA 94305-9010, USA, E-mail: khatib@cs.stanford.edu

Professor Frans Groen, Department of Computer Science, Universiteit van Amsterdam, Kruislaan 403,
1098 SJ Amsterdam, The Netherlands, E-mail: groen@science.uva.nl

Author

Peter Corke

Faculty of Built Environment and Engineering
School of Engineering Systems
Queensland University of Technology (QUT)
Brisbane QLD 4000
Australia
e-mail: rvc@petercorke.com

ISBN 978-3-642-20143-1

e-ISBN 978-3-642-20144-8

DOI 10.1007/978-3-642-20144-8

Springer Tracts in Advanced Robotics

ISSN 1610-7438

Library of Congress Control Number: 2011934624

© Springer-Verlag Berlin Heidelberg 2011

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitations, broadcasting, reproduction on microfilm or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer. Violations are liable to prosecution under the German Copyright Law.

The use of general descriptive names, registered names, trademarks, 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.

Production: Armin Stasch and Scientific Publishing Services Pvt. Ltd. Chennai, India
Typesetting and layout: Büro Stasch · Bayreuth (stasch@stasch.com)

Printed on acid-free paper

9 8 7 6 5 4 3 2 1

springer.com

Editorial Advisory Board

Oliver Brock, TU Berlin, Germany
Herman Bruyninckx, KU Leuven, Belgium
Raja Chatila, LAAS, France
Henrik Christensen, Georgia Tech, USA
Peter Corke, Queensland Univ. Technology, Australia
Paolo Dario, Scuola S. Anna Pisa, Italy
Rüdiger Dillmann, Univ. Karlsruhe, Germany
Ken Goldberg, UC Berkeley, USA
John Hollerbach, Univ. Utah, USA
Makoto Kaneko, Osaka Univ., Japan
Lydia Kavraki, Rice Univ., USA
Vijay Kumar, Univ. Pennsylvania, USA
Sukhan Lee, Sungkyunkwan Univ., Korea
Frank Park, Seoul National Univ., Korea
Tim Salcudean, Univ. British Columbia, Canada
Roland Siegwart, ETH Zurich, Switzerland
Gaurav Sukhatme, Univ. Southern California, USA
Sebastian Thrun, Stanford Univ., USA
Yangsheng Xu, Chinese Univ. Hong Kong, PRC
Shin'ichi Yuta, Tsukuba Univ., Japan

STAR (Springer Tracts in Advanced Robotics) has been promoted under the auspices of EURON (European Robotics Research Network)



*To my family Phillipa, Lucy and Madeline for their indulgence and support;
my parents Margaret and David for kindling my curiosity;
and to Lou Paul who planted the seed that became this book.*

Foreword

Once upon a time, a very thick document of a dissertation from a faraway land came to me for evaluation. *Visual robot control* was the thesis theme and *Peter Corke* was its author. Here, I am reminded of an excerpt of my comments, which reads, *this is a masterful document, a quality of thesis one would like all of one's students to strive for, knowing very few could attain – very well considered and executed.*

The connection between robotics and vision has been, for over two decades, the central thread of Peter Corke's productive investigations and successful developments and implementations. This rare experience is bearing fruit in his new book on *Robotics, Vision, and Control*. In its melding of theory and application, this new book has considerably benefited from the author's unique mix of academic and real-world application influences through his many years of work in robotic mining, flying, underwater, and field robotics.

There have been numerous textbooks in robotics and vision, but few have reached the level of integration, analysis, dissection, and practical illustrations evidenced in this book. The discussion is thorough, the narrative is remarkably informative and accessible, and the overall impression is of a significant contribution for researchers and future investigators in our field. Most every element that could be considered as relevant to the task seems to have been analyzed and incorporated, and the effective use of Toolbox software echoes this thoroughness.

The reader is taken on a realistic walkthrough the fundamentals of mobile robots, navigation, localization, manipulator-arm kinematics, dynamics, and joint-level control, as well as camera modeling, image processing, feature extraction, and multi-view geometry. These areas are finally brought together through extensive discussion of visual servo system. In the process, the author provides insights into how complex problems can be decomposed and solved using powerful numerical tools and effective software.

The *Springer Tracts in Advanced Robotics (STAR)* is devoted to bringing to the research community the latest advances in the robotics field on the basis of their significance and quality. Through a wide and timely dissemination of critical research developments in robotics, our objective with this series is to promote more exchanges and collaborations among the researchers in the community and contribute to further advancements in this rapidly growing field.

Peter Corke brings a great addition to our STAR series with an authoritative book, reaching across fields, thoughtfully conceived and brilliantly accomplished.

Oussama Khatib
Stanford, California
July 2011

Preface

*Tell me and I will forget.
Show me and I will remember.
Involve me and I will understand.*
Chinese proverb

The practice of robotics and machine vision involves the application of computational algorithms to data. The data comes from sensors measuring the velocity of a wheel, the angle of a robot arm's joint or the intensities of millions of pixels that comprise an image of the world that the robot is observing. For many robotic applications the amount of data that needs to be processed, in real-time, is massive. For vision it can be of the order of tens to hundreds of megabytes per second.

Progress in robots and machine vision has been, and continues to be, driven by more effective ways to process data. This is achieved through new and more efficient algorithms, and the dramatic increase in computational power that follows Moore's law. When I started in robotics and vision, in the mid 1980s, the IBM PC had been recently released – it had a 4.77 MHz 16-bit microprocessor and 16 kbytes (expandable to 256 k) of memory. Over the intervening 25 years computing power has doubled 16 times which is an increase by a factor of 65 000. In the late 1980s systems capable of real-time image processing were large 19 inch racks of equipment such as shown in Fig. 0.1. Today there is far more computing in just a small corner of a modern microprocessor chip.

Over the fairly recent history of robotics and machine vision a very large body of algorithms has been developed – a significant, tangible, and collective achievement of the research community. However its sheer size and complexity presents a barrier to somebody entering the field. Given the many algorithms from which to choose the obvious question is:

What is the right algorithm for this particular problem?

One strategy would be to try a few different algorithms and see which works best for the problem at hand but this raises the next question:

How can I evaluate algorithm X on my own data without spending days coding and debugging it from the original research papers?



Fig. 0.1.

Once upon a time a lot of equipment was needed to do vision-based robot control. The author with a large rack full of image processing and robot control equipment (1992)

Two developments come to our aid. The first is the availability of general purpose mathematical software which makes it easy to prototype algorithms. There are commercial packages such as MATLAB®, Mathematica and MathCad, and open source projects include SciLab, Octave, and PyLab. All these tools deal naturally and effortlessly with vectors and matrices, can create complex and beautiful graphics, and can be used interactively or as a programming environment. The second is the open-source movement. Many algorithms developed by researchers are available in open-source form. They might be coded in one of the general purpose mathematical languages just mentioned, or written in a mainstream language like C, C++ or Java.

For more than fifteen years I have been part of the open-source community and maintained two open-source MATLAB® Toolboxes: one for robotics and one for machine vision. They date back to my own PhD work and have evolved since then, growing features and tracking changes to the MATLAB® language (which have been significant over that period). The Robotics Toolbox has also been translated into a number of different languages such as Python, SciLab and LabView.

The Toolboxes have some important virtues. Firstly, they have been around for a long time and used by many people for many different problems so the code is entitled to some level of trust. The Toolbox provides a “gold standard” with which to compare new algorithms or even the same algorithms coded in new languages or executing in new environments.

Secondly, they allow the user to work with real problems, not trivial examples. For real robots, those with more than two links, or real images with millions of pixels the computation is beyond unaided human ability. Thirdly, they allow us to gain insight which is otherwise lost in the complexity. We can rapidly and easily experiment, play *what if* games, and depict the results graphically using MATLAB®’s powerful display tools such as 2D and 3D graphs and images.

Fourthly, the Toolbox code makes many common algorithms tangible and accessible. You can read the code, you can apply it to your own problems, and you can extend it or rewrite it. At the very least it gives you a headstart.

The Toolboxes were always accompanied by short tutorials as well as reference material. Over the years many people have urged me to turn this into a book and finally it has happened! The purpose of this book is to expand on the tutorial material provided with the Toolboxes, add many more examples, and to weave it into a narrative that covers robotics and computer vision separately and together. I want to show how complex problems can be decomposed and solved using just a few simple lines of code.

By inclination I am a *hands on* person. I like to program and I like to analyze data, so it has always seemed natural to me to build tools to solve problems in robotics and vision. The topics covered in this book are based on my own interests but also guided by real problems that I observed over many years as a practitioner of both robotics and computer vision. I hope that by the end of this book you will share my enthusiasm for these topics.

I was particularly motivated to present a solid introduction to machine vision for roboticists. The treatment of vision in robotics textbooks tends to concentrate on simple binary vision techniques. In the book we will cover a broad range of topics including color vision, advanced segmentation techniques such as maximally stable extremal regions and graphcuts, image warping, stereo vision, motion estimation and image retrieval. We also cover non-perspective imaging using fisheye lenses and catadioptric optics. These topics are growing in importance for robotics but are not commonly covered. Vision is a powerful sensor, and roboticists should have a solid grounding in modern fundamentals. The last part of the book shows how vision can be used as the primary sensor for robot control.

This book is unlike other text books, and deliberately so. Firstly, there are already a number of excellent text books that cover robotics and computer vision separately and in depth, but few that cover both in an integrated fashion. Achieving this integration is a principal goal of this book.

Respectively the trademarks of
The Mathworks Inc., Wolfram
Research, and PTC.

Secondly, software is a first-class citizen in this book. Software is a tangible instantiation of the algorithms described – it can be read and it can be pulled apart, modified and put back together again. There are a number of classic books that use software in this illustrative fashion for problem solving. In this respect I’ve been influenced by books such as *LaTeX: A document preparation system* (Lamport 1994), *Numerical Recipes in C* (Press et al. 2007), *The Little Lisper* (Friedman et al. 1987) and *Structure and Interpretation of Classical Mechanics* (Sussman et al. 2001). The many examples in this book illustrate how the Toolbox software can be used and generally provide *instant gratification* in just a couple of lines of MATLAB® code.

Thirdly, building the book around MATLAB® and the Toolboxes means that we are able to tackle more realistic and more complex problems than other books.

The emphasis on software and examples does not mean that rigour and theory are unimportant, they are very important, but this book provides a complementary approach. It is best read in conjunction with standard texts which provide rigour and theoretical nourishment. The end of each chapter has a section on further reading and provides pointers to relevant textbooks and key papers.

Writing this book provided a good opportunity to look critically at the Toolboxes and to revise and extend the code. In particular I’ve made much greater use of the ever-evolving object-oriented features of MATLAB® to simplify the user interface and to reduce the number of separate files within the Toolboxes.

The rewrite also made me look more widely at complementary open-source code. There is a lot of great code out there, particularly on the computer vision side, so rather than reinvent some wheels I’ve tried to integrate the best code I could find for particular algorithms. The complication is that every author has their own naming conventions and preferences about data organization, from simple matters like the use of row or column vectors to more complex issues involving structures – arrays of structures or structures of arrays. My solution has been, as much as possible, to not modify any of these packages but to encapsulate them with light weight wrappers, particularly as classes.

I am grateful to the following for code that has been either incorporated into the Toolboxes or which has been wrapped into the Toolboxes. Robotics Toolbox contributions include: mobile robot localization and mapping by Paul Newman at Oxford and a quadcopter simulator by Paul Pounds at Yale. Machine Vision Toolbox contributions include: RANSAC code by Peter Kovesi; pose estimation by Francesco Moreno-Noguer, Vincent Lepetit, Pascal Fua at the CVLab-EPFL; color space conversions by Pascal Getreuer; numerical routines for geometric vision by various members of the Visual Geometry Group at Oxford (from the web site of the Hartley and Zisserman book; Hartley and Zisserman 2003); the k -means and MSER algorithms by Andrea Vedaldi and Brian Fulkerson; the graph-based image segmentation software by Pedro Felzenszwalb; and the SURF feature detector by Dirk-Jan Kroon at U. Twente. The Camera Calibration Toolbox by Jean-Yves Bouguet is used unmodified.

Along the way I got interested in the mathematicians, physicists and engineers whose work, hundreds of years later, is critical to the science of robotic and vision today. Some of their names have become adjectives like Coriolis, Gaussian, Laplacian or Cartesian; nouns like Jacobian, or units like Newton and Coulomb. They are interesting characters from a distant era when science was a hobby and their day jobs were as doctors, alchemists, gamblers, astrologers, philosophers or mercenaries. In order to know whose shoulders we are standing on I have included small vignettes about the lives of these people – a smattering of history as a backstory.

In my own career I have had the good fortune to work with many wonderful people who have inspired and guided me. Long ago at the University of Melbourne John Anderson fired my interest in control and Graham Holmes encouraged me to “think before I code” – excellent advice that I sometimes heed. Early on I spent a life-direction-changing ten months working with Richard (Lou) Paul in the GRASP laboratory at the University of Pennsylvania in the period 1988–1989. The genesis of the Toolboxes was my

PhD research and my advisors Malcolm Good (University of Melbourne) and Paul Dunn (CSIRO) asked me good questions and guided my research. Laszlo Nemes provided sage advice about life and the ways of organizations and encouraged me to publish more and to open-source my software. Much of my career was spent at CSIRO where I had the privilege and opportunity to work on a diverse range of real robotics projects and to work with a truly talented set of colleagues and friends. Mid book I joined Queensland University of Technology which has generously made time available to me to complete the project. My former students Jasmine Banks, Kane Usher, Paul Pounds and Peter Hansen taught me a lot of about stereo, non-holonomy, quadcopters and wide-angle vision respectively.

I would like to thank Paul Newman for generously hosting me several times at Oxford where significant sections of the book were written, and Daniela Rus for hosting me at MIT for a burst of intense writing that was the first complete book draft. Daniela, Paul and Cédric Pradalier made constructive suggestions and comments on early drafts of the material. I would also like to thank the MathWorks, the publishers of MATLAB® for the support they offered me through their author program. Springer have been enormously supportive of the whole project and a pleasure to work with. I would specially like to thank Thomas Ditzinger, my editor, and Armin Stasch for the layout and typesetting which has transformed my manuscript into a book.

I have tried my hardest to eliminate errors but inevitably some will remain. Please email me bug reports as well as suggestions for improvements and extensions.

Finally, it can't be easy living with a writer – there are books and websites devoted to this topic. My deepest thanks are to Phillipa for supporting and encouraging me in the endeavour and living with “the book” for so long and in so many different places.

Peter Corke
Brisbane, Queensland
June 2011

Contents

1	Introduction	1
1.1	About the Book	6
1.1.1	The MATLAB Software	7
1.1.2	Audience and Prerequisites	8
1.1.3	Notation and Conventions	9
1.1.4	How to Use the Book	9
1.1.5	Teaching with the Book	10
1.1.6	Outline	10
	Part I Foundations	13
2	Representing Position and Orientation	15
2.1	Representing Pose in 2-Dimensions	19
2.2	Representing Pose in 3-Dimensions	24
2.2.1	Representing Orientation in 3-Dimensions	25
2.2.2	Combining Translation and Orientation	37
2.3	Wrapping Up	39
	Further Reading	40
	Exercises	41
3	Time and Motion	43
3.1	Trajectories	43
3.1.1	Smooth One-Dimensional Trajectories	43
3.1.2	Multi-Dimensional Case	46
3.1.3	Multi-Segment Trajectories	46
3.1.4	Interpolation of Orientation in 3D	48
3.1.5	Cartesian Motion	49
3.2	Time Varying Coordinate Frames	51
3.2.1	Rotating Coordinate Frame	51
3.2.2	Incremental Motion	52
3.2.3	Inertial Navigation Systems	53
3.3	Wrapping Up	56
	Further Reading	56
	Exercises	56
	Part II Mobile Robots	59
4	Mobile Robot Vehicles	65
4.1	Mobility	65
4.2	Car-like Mobile Robots	67
4.2.1	Moving to a Point	71
4.2.2	Following a Line	72
4.2.3	Following a Path	74
4.2.4	Moving to a Pose	75

4.3	Flying Robots	78
4.4	Wrapping Up	84
	Further Reading	84
	Exercises	85
5	Navigation	87
5.1	Reactive Navigation	88
5.1.1	Braitenberg Vehicles	88
5.1.2	Simple Automata	90
5.2	Map-Based Planning	91
5.2.1	Distance Transform	93
5.2.2	D*	95
5.2.3	Voronoi Roadmap Method	97
5.2.4	Probabilistic Roadmap Method	99
5.2.5	RRT	102
5.3	Wrapping Up	104
	Further Reading	105
	Exercises	106
6	Localization	107
6.1	Dead Reckoning	111
6.1.1	Modeling the Vehicle	111
6.1.2	Estimating Pose	113
6.2	Using a Map	116
6.3	Creating a Map	120
6.4	Localization and Mapping	123
6.5	Monte-Carlo Localization	125
6.6	Wrapping Up	128
	Further Reading	129
	Notes on Toolbox Implementation	130
	Exercises	130
	Part III Arm-Type Robots	133
7	Robot Arm Kinematics	137
7.1	Describing a Robot Arm	137
7.2	Forward Kinematics	140
7.2.1	A 2-Link Robot	141
7.2.2	A 6-Axis Robot	143
7.3	Inverse Kinematics	146
7.3.1	Closed-Form Solution	146
7.3.2	Numerical Solution	149
7.3.3	Under-Actuated Manipulator	149
7.3.4	Redundant Manipulator	150
7.4	Trajectories	152
7.4.1	Joint-Space Motion	153
7.4.2	Cartesian Motion	155
7.4.3	Motion through a Singularity	156
7.4.4	Configuration Change	157
7.5	Advanced Topics	158
7.5.1	Joint Angle Offsets	158
7.5.2	Determining Denavit-Hartenberg Parameters	159
7.5.3	Modified Denavit-Hartenberg Notation	160
7.6	Application: Drawing	162

7.7	Application: a Simple Walking Robot	163
7.7.1	Kinematics	163
7.7.2	Motion of One Leg	165
7.7.3	Motion of Four Legs	166
7.8	Wrapping Up	167
	Further Reading	168
	The plot Method	168
	Exercises	170
8	Velocity Relationships	171
8.1	Manipulator Jacobian	171
8.1.1	Transforming Velocities between Coordinate Frames	174
8.1.2	Jacobian in the End-Effector Coordinate Frame	175
8.1.3	Analytical Jacobian	176
8.1.4	Jacobian Condition and Manipulability	177
8.2	Resolved-Rate Motion Control	180
8.2.1	Jacobian Singularity	182
8.2.2	Jacobian for under-Actuated Robot	183
8.2.3	Jacobian for over-Actuated Robot	184
8.3	Force Relationships	186
8.3.1	Transforming Wrenches between Frames	186
8.3.2	Transforming Wrenches to Joint Space	186
8.4	Inverse Kinematics: a General Numerical Approach	187
8.5	Wrapping Up	188
	Further Reading	189
	Exercises	189
9	Dynamics and Control	191
9.1	Equations of Motion	191
9.1.1	Gravity Term	193
9.1.2	Inertia Matrix	195
9.1.3	Coriolis Matrix	196
9.1.4	Effect of Payload	197
9.1.5	Base Force	198
9.1.6	Dynamic Manipulability	198
9.2	Drive Train	200
9.2.1	Friction	201
9.3	Forward Dynamics	202
9.4	Manipulator Joint Control	204
9.4.1	Actuators	204
9.4.2	Independent Joint Control	204
9.4.3	Rigid-Body Dynamics Compensation	211
9.4.4	Flexible Transmission	213
9.5	Wrapping Up	215
	Further Reading	216
	Exercises	217
	Part IV Computer Vision	219
10	Light and Color	223
10.1	Spectral Representation of Light	223
10.1.1	Absorption	225
10.1.2	Reflection	226

10.2	Color	227
10.2.1	Reproducing Colors	230
10.2.2	Chromaticity Space	233
10.2.3	Color Names	236
10.2.4	Other Color Spaces	236
10.2.5	Transforming between Different Primaries	238
10.2.6	What Is White?	240
10.3	Advanced Topics	240
10.3.1	Color Constancy	241
10.3.2	White Balancing	241
10.3.3	Color Change Due to Absorption	242
10.3.4	Gamma	243
10.3.5	Application: Color Image	245
10.4	Wrapping Up	247
	Further Reading	248
	Data Sources	249
	Exercises	249
11	Image Formation	251
11.1	Perspective Transform	251
11.1.1	Lens Distortion	261
11.2	Camera Calibration	262
11.2.1	Homogeneous Transformation Approach	262
11.2.2	Decomposing the Camera Calibration Matrix	264
11.2.3	Pose Estimation	266
11.2.4	Camera Calibration Toolbox	266
11.3	Non-Perspective Imaging Models	269
11.3.1	Fisheye Lens Camera	270
11.3.2	Catadioptric Camera	272
11.3.3	Spherical Camera	274
11.4	Unified Imaging	275
11.4.1	Mapping Wide-Angle Images to the Sphere	276
11.4.2	Synthetic Perspective Images	278
11.5	Wrapping Up	280
	Further Reading	280
	Camera Classes	282
	Exercises	283
12	Image Processing	285
12.1	Obtaining an Image	285
12.1.1	Images from Files	285
12.1.2	Images from an Attached Camera	289
12.1.3	Images from a Movie File	289
12.1.4	Images from the Web	290
12.1.5	Images from Code	291
12.2	Monadic Operations	293
12.3	Diadic Operations	296
12.4	Spatial Operations	299
12.4.1	Convolution	300
12.4.2	Template Matching	311
12.4.3	Non-Linear Operations	316
12.5	Mathematical Morphology	317
12.5.1	Noise Removal	321

12.5.2	Boundary Detection	322
12.5.3	Hit and Miss Transform	322
12.6	Shape Changing	324
12.6.1	Cropping	324
12.6.2	Image Resizing	324
12.6.3	Image Pyramids	326
12.6.4	Image Warping	327
12.7	Wrapping Up	330
	Further Reading	330
	Sources of Image Data	332
	MATLAB® Software Tools	332
	General Software Tools	332
	Exercises	333
13	Image Feature Extraction	335
13.1	Region Features	337
13.1.1	Classification	337
13.1.2	Representation	346
13.1.3	Description	350
13.1.4	Recap	360
13.2	Line Features	361
13.3	Point Features	365
13.3.1	Classical Corner Detectors	366
13.3.2	Scale-Space Corner Detectors	371
13.4	Wrapping Up	376
	Further Reading	376
	Exercises	378
14	Using Multiple Images	381
14.1	Feature Correspondence	382
14.2	Geometry of Multiple Views	386
14.2.1	The Fundamental Matrix	388
14.2.2	The Essential Matrix	390
14.2.3	Estimating the Fundamental Matrix	391
14.2.4	Planar Homography	396
14.3	Stereo Vision	401
14.3.1	Sparse Stereo	401
14.3.2	Dense Stereo Matching	405
14.3.3	Peak Refinement	412
14.3.4	Cleaning up and Reconstruction	413
14.3.5	3D Texture Mapped Display	415
14.3.6	Anaglyphs	416
14.3.7	Image Rectification	417
14.3.8	Plane Fitting	419
14.3.9	Matching Sets of 3D Points	420
14.4	Structure and Motion	422
14.5	Application: Perspective Correction	428
14.6	Application: Mosaicing	431
14.7	Application: Image Matching and Retrieval	433
14.8	Application: Image Sequence Processing	439
14.9	Wrapping Up	442
	Further Reading	442
	Resources	445
	Exercises	446

Part V Robotics, Vision and Control	451
15 Vision-Based Control	455
15.1 Position-Based Visual Servoing	456
15.2 Image-Based Visual Servoing	459
15.2.1 Camera and Image Motion	460
15.2.2 Controlling Feature Motion	464
15.2.3 Depth	469
15.2.4 Performance Issues	471
15.3 Using Other Image Features	473
15.3.1 Line Features	473
15.3.2 Circle Features	474
15.4 Wrapping Up	476
Further Reading	476
Exercises	478
16 Advanced Visual Servoing	481
16.1 XY/Z-Partitioned IBVS	481
16.2 IBVS Using Polar Coordinates	484
16.3 IBVS for a Spherical Camera	486
16.4 Application: Arm-Type Robot	488
16.5 Application: Mobile Robot	489
16.5.1 Holonomic Mobile Robot	489
16.5.2 Non-Holonomic Mobile Robot	491
16.6 Application: Aerial Robot	492
16.7 Wrapping Up	494
Further Reading	494
Exercises	495
Appendices	497
A Installing the Toolboxes	499
B Simulink®	501
C MATLAB® Objects	505
D Linear Algebra Refresher	511
E Ellipses	517
F Gaussian Random Variables	523
G Jacobians	527
H Kalman Filter	529
I Homogeneous Coordinates	533
J Graphs	535
K Peak Finding	539
Bibliography	543
Index	553
Index of People	553
Index of Functions, Classes and Methods	554
General Index	558

Nomenclature

The notation used in robotics and computer vision varies considerably from book to book. The symbols used in this book, and their units where appropriate, are listed below. Some symbols have multiple meanings and their context must be used to disambiguate them.

The elements of a vector $\mathbf{x}[i]$ or a matrix $\mathbf{x}[i, j]$ are indicated by square brackets. The elements of a time series $\mathbf{x}\langle k \rangle$ are indicated by angle brackets.

Symbol	Description	Unit
\hat{x}	an estimate of x	
\bar{x}	mean of x or relative value	
x^*	desired value of x	
\mathbf{v}	a vector	
$\hat{\mathbf{v}}$	a unit-vector parallel to \mathbf{v}	
$ \mathbf{v} $	scalar norm or length of the vector \mathbf{v}	
$ \hat{q} $	scalar norm of the quaternion \hat{q}	
$\tilde{\mathbf{v}}$	homogeneous representation of vector \mathbf{v}	
v_x	a component of a vector	
$\mathbf{v}_1 \cdot \mathbf{v}_2$	dot, or inner, product, also $\mathbf{v}_1^T \mathbf{v}_2$	
$\mathbf{v}_1 \times \mathbf{v}_2$	cross, or vector, product	
\mathbf{A}	a matrix	
\mathbf{A}^{-1}	inverse of \mathbf{A}	
\mathbf{A}^+	pseudo-inverse of \mathbf{A}	
\mathbf{A}^T	transpose of \mathbf{A}	
\mathbf{A}^{-T}	transpose of inverse \mathbf{A}	
$A_{i,j}$	the element (i, j) of \mathbf{A}	
$\mathbf{A}[i, j]$	the element (i, j) of \mathbf{A}	
$F(x)$	a function of x	
$F_x(x)$	the derivative $\partial F / \partial x$	
B	viscous friction coefficient	N m s rad ⁻¹
\mathcal{C}	configuration space of a robot	
\mathbf{C}	camera matrix, $\mathbf{C} \in \mathbb{R}^{3 \times 4}$	
$\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$	manipulator centripetal and Coriolis term	kg m ² s ⁻¹
\mathbb{C}	the set of complex numbers	
$\mathcal{D}(\cdot)$	manipulator dynamics function: $\mathbf{Q}, \mathbf{q}, \dot{\mathbf{q}} \mapsto \ddot{\mathbf{q}}$	
$\Delta(\xi)$	maps incremental pose change to differential motion: $SE(3) \mapsto \mathbb{R}^6$	
$\Delta^{-1}(\delta)$	maps differential motion to incremental pose change: $\mathbb{R}^6 \mapsto SE(3)$	
E	illuminance (lux)	lx
f	focal length	m
f	force	N
\mathbf{f}	vector of image features	

Symbol	Description	Unit
$F(\dot{q})$	friction torque	N m
ϕ	luminous flux (lumens)	lm
g	wrench, a vector of forces and moments $(f_x, f_y, f_z, m_x, m_y, m_z)$	N, Nm
$G(q)$	manipulator gravity loading term	N m
γ	robot steering angle	rad
Γ	3-angle representation of rotation, $\Gamma \in \mathbb{R}^3$	rad
Γ	body torque $\Gamma \in \mathbb{R}^3$	N m
$I_{n \times n}$	$n \times n$ identity matrix	
J	inertia	kg m ²
J	inertia tensor, 3×3 matrix	kg m ²
J	Jacobian matrix	
${}^A J_B$	Jacobian transforming velocities in frame A to frame B	
k, K	constant	
K	camera calibration matrix	
K_i	amplifier gain (transconductance)	A V ⁻¹
K_m	motor torque constant	N m A ⁻¹
$\mathcal{K}(\cdot)$	forward kinematics	
$\mathcal{K}^{-1}(\cdot)$	inverse kinematics	
L	luminance (nit)	nt
λ	wavelength	m
λ	an eigenvalue	
m_i	mass of link i	kg
$M(q)$	manipulator inertia matrix	kg m ²
\mathbf{p}	an image plane point	
\mathbf{P}	a world point	
\mathbb{P}^2	the projective space of all 2-D points, a 3-tuple	
\mathbb{P}^3	the projective space of all 3-D points, a 4-tuple	
$\mathcal{P}(\cdot)$	projection function: $\mathbb{R}^3 \mapsto \mathbb{R}^2$	
\hat{q}	quaternion	
$\hat{q}(v)$	pure quaternion of vector v	
q	configuration, generalized coordinates	m, rad
Q	generalized force	N, Nm
ρ_w, ρ_h	pixel width and height	m
R	an orthonormal rotation matrix, $R \in SO(2)$ or $SO(3)$	
\mathbb{R}	set of real numbers	
\mathbb{R}^2	the space of all 2-D points	
\mathbb{R}^3	the space of all 3-D points	
s	Laplace transform operator	
\mathbb{S}	set of all angles in the circle $[0, 2\pi)$	
$SE(n)$	special Euclidean group (all poses) in n dimensions	
$SO(n)$	special orthogonal group, the set of all orientations in n dimensions	
$S(v)$	skew symmetric matrix of v	
s_i	COM of link i with respect to the link i coordinate frame	m
S_i	first moment of link i . $S_i = m_i s_i$	kg m
σ	standard deviation	
σ	robot joint type, $\sigma = 0$ for revolute and $\sigma = 1$ for prismatic	
t	time	s

Symbol	Description	Unit
T	sample interval	s
T	temperature	K
T	optical transmission	
T	homogeneous transformation, $T \in SE(2)$ or $SE(3)$	
${}^A T_B$	homogeneous transform representing frame $\{B\}$ with respect to frame $\{A\}$. If A is not given then assumed relative to world coordinate frame 0. Note that ${}^A T_B = ({}^B T_A)^{-1}$	
θ	angle	rad
$\boldsymbol{\theta}$	vector of angles, generally robot joint angles	rad
$\theta_r, \theta_p, \theta_y$	roll pitch yaw angles	rad
τ	torque	N m
τ_C	Coulomb friction torque	N m
u, v	camera image plane coordinates	pixels
\bar{u}, \bar{v}	normalized image plane coordinates, relative to the principal point	m
u_0, v_0	coordinates of the principal point	pixels
v	velocity	m s ⁻¹
\mathbf{v}	velocity vector	m s ⁻¹
$\boldsymbol{\nu}$	innovation	
$\boldsymbol{\nu}$	velocity screw, $\boldsymbol{\nu} \in \mathbb{R}^6$, $\boldsymbol{\nu} = (v_x, v_y, v_z, \omega_x, \omega_y, \omega_z)$	
ω	rotational rate	rad s ⁻¹
$\boldsymbol{\omega}$	angular velocity vector	rad s ⁻¹
X, Y, Z	Cartesian coordinates	
ξ	abstract representation of 3-dimensional Cartesian pose (pronounced ksi)	
${}^A \xi_B$	abstract representation of 3-dimensional relative pose, frame $\{B\}$ with respect to frame $\{A\}$	
$\boldsymbol{\nu}$	Cartesian velocity screw $(v_x, v_y, v_z, \omega_x, \omega_y, \omega_z)$	
\bar{x}, \bar{y}	normalized image-plane coordinates	
$\mathbf{0}_{m \times n}$	an $m \times n$ matrix of zeros	
$\mathbf{1}_{m \times n}$	an $m \times n$ matrix of ones	
\mathbb{Z}	the set of all integers	
\mathbb{Z}^+	the set of all integers greater than zero	
\sim	equivalence of representations	
\simeq	homogeneous coordinate equivalence	
\oplus	pose composition operator	
\equiv	colormetric equivalence	
\ominus	inverse of a pose (unary operator)	
\cdot	transformation of a point by a relative pose, e.g. $\xi \cdot p$	
\ominus	smallest angular difference on a circle	rad
\otimes	convolution	
\oplus	morphological dilation	
\ominus	morphological erosion	
\circ	morphological opening	
\bullet	morphological closing	
$\{F\}$	coordinate frame F	
$[a, b]$	interval a to b inclusive	
(a, b)	interval a to b , not including a or b	
$[a, b)$	interval a to b , not including b	
$(a, b]$	interval a to b exclusive, not including a	

MATLAB® Toolbox Conventions

- A Cartesian coordinate, a point, is expressed as a column vector.
- A set of points is expressed as a matrix with columns representing the coordinates of individual points.
- A robot configuration, a set of joint angles, is expressed as a row vector.
- Time series data is expressed as a matrix with rows representing time steps.