Springer Tracts in Advanced Robotics
Volume 73

Peter Corke

Robotics, Vision and Control

Fundamental Algorithms in MATLAB®

With 393 Images

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

Springer

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

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

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 it 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.

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

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.

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

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 $\boldsymbol{x}[i]$ or a matrix $\boldsymbol{x}[i, j]$ are indicated by square brackets. The elements of a time series $\boldsymbol{x}\langle k \rangle$ 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	
$\boldsymbol{v}$	a vector	
$\hat{\boldsymbol{v}}$	a unit-vector parallel to $\boldsymbol{v}$	
$\lvert\boldsymbol{v}\rvert$	scalar norm or length of the vector $\boldsymbol{v}$	
$\lvert\mathring{q}\rvert$	scalar norm of the quaternion $\mathring{q}$	
$\tilde{\boldsymbol{v}}$	homogeneous representation of vector $\boldsymbol{v}$	
v_x	a component of a vector	
$\boldsymbol{v}_1 \cdot \boldsymbol{v}_2$	dot, or inner, product, also $\boldsymbol{v}_1^T \boldsymbol{v}_2$	
$\boldsymbol{v}_1 \times \boldsymbol{v}_2$	cross, or vector, product	
$\boldsymbol{A}$	a matrix	
$\boldsymbol{A}^{-1}$	inverse of $\boldsymbol{A}$	
$\boldsymbol{A}^+$	pseudo-inverse of $\boldsymbol{A}$	
$\boldsymbol{A}^T$	transpose of $\boldsymbol{A}$	
$\boldsymbol{A}^{-T}$	transpose of inverse $\boldsymbol{A}$	
$\boldsymbol{A}_{i,j}$	the element (i, j) of $\boldsymbol{A}$	
$\boldsymbol{A}[i, j]$	the element (i, j) of $\boldsymbol{A}$	
$F(x)$	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	
$\boldsymbol{C}$	camera matrix, $\boldsymbol{C} \in \mathbb{R}^{3\times4}$	
$\boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})$	manipulator centripetal and Coriolis term	kg m^2 s^{-1}
$\mathbb{C}$	the set of complex numbers	
$\mathcal{D}(\cdot)$	manipulator dynamics function: $\boldsymbol{Q}, \boldsymbol{q}, \dot{\boldsymbol{q}} \mapsto \ddot{\boldsymbol{q}}$	
$\Delta(\xi)$	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
$\boldsymbol{f}$	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
$\boldsymbol{G}(\boldsymbol{q})$	manipulator gravity loading term	N m
γ	robot steering angle	rad
$\boldsymbol{\Gamma}$	3-angle representation of rotation, $\boldsymbol{\Gamma} \in \mathbb{R}^3$	rad
$\boldsymbol{\Gamma}$	body torque $\boldsymbol{\Gamma} \in \mathbb{R}^3$	N m
$\boldsymbol{I}_{n\times n}$	$n \times n$ identity matrix	
J	inertia	kg m^2
$\boldsymbol{J}$	inertia tensor, 3×3 matrix	kg m^2
$\boldsymbol{J}$	Jacobian matrix	
${}^A\boldsymbol{J}_B$	Jacobian transforming velocities in frame A to frame B	
k, K	constant	
$\boldsymbol{K}$	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
$\mathbf{M}(\boldsymbol{q})$	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: $\mathbb{R}^3 \mapsto \mathbb{R}^2$	
$\mathring{q}$	quaternion	
$\mathring{q}(\boldsymbol{v})$	pure quaternion of vector $\boldsymbol{v}$	
$\boldsymbol{q}$	configuration, generalized coordinates	m, rad
$\boldsymbol{Q}$	generalized force	N, Nm
ρ_w, ρ_h	pixel width and height	m
$\boldsymbol{R}$	an orthonormal rotation matrix, $\boldsymbol{R} \in SO(2)$ 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	
$\boldsymbol{S}(\boldsymbol{v})$	skew symmetric matrix of $\boldsymbol{v}$	
$\boldsymbol{s}_i$	COM of link i with respect to the link i coordinate frame	m
$\boldsymbol{S}_i$	first moment of link i. $\boldsymbol{S}_i = m_i\, \boldsymbol{s}_i$	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	
$\boldsymbol{T}$	homogeneous transformation, $T \in SE(2)$ or $SE(3)$	
${}^{A}\boldsymbol{T}_{B}$	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 ${}^{A}\boldsymbol{T}_{B} = ({}^{B}\boldsymbol{T}_{A})^{-1}$	
θ	angle	rad
$\boldsymbol{\theta}$	vector of angles, generally robot joint angles	rad
θ_r, θ_p, θ_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}
$\boldsymbol{v}$	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. $\xi \cdot \boldsymbol{p}$	
$\ominus$	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.